НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:

В.о. завідувача кафедри

_____ Оксана ТИМОЩУК

«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи та методи штучного інтелекту»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Автоматизована система для розробки стратегії просування у сфері інтернет-маркетингу»

Виконав:

студент IV курсу, групи КА-65
Терещенко Антон В'ячеславович _____

Керівник:
Ассистент Древаль Максим Михайлович _____

Консультант з нормоконтролю:
Доцент, к.т.н. Коваленко Анатолій Єпіванович _____

Консультант з економічного розділу:
Доцент, к.е.н. Шевчук Олена Анатоліївна _____

Рецензент:
Співзасновниця рекламного агенства 8tag _____
Махаєва Катерина Олексіївна

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»
Освітньо-професійна програма «Системи та методи штучного інтелекту»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Оксана ТИМОЩУК
«25» травня 2020 р.

ЗАВДАННЯ

на дипломну роботу студенту
Терещенку Антону В'ячеславовичу

1. Тема роботи «Автоматизована система для розробки стратегії просування у сфері інтернет-маркетингу», керівник роботи Древаль Максим Михайлович, асистент, затверджені наказом по університету від «25» травня 2020 р. № 1143-с

2. Термін подання студентом роботи 8.06.2020.

3. Вихідні дані до роботи: статистичні дані завершених рекламних кампаній, типи моделей, критерії адекватностей моделей, оцінювання якості прогнозу.

4. Зміст роботи: дослідження та особливості предметної області систем інтернет-маркетингу, процес та підходи до оптимізації показників просування.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо):

(1) мета та цілі;

(2) актуальність;

(3) етапи роботи програми;

(4) підхід до оптимізації;

(5) фактичний аналіз прогнозування.

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Коваленко А. Є., доцент		
Економічний	Шевчук О. А., доцент		

7. Дата видачі завдання 17 жовтня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Затвердження теми БДР	14.10.2019-21.10.2019	
2.	Ознайомлення зі структурою БДР згідно з Положенням про державну атестацію студентів НТУУ «КПІ»	21.10.2019-04.11.2019	
3.	Ознайомлення з ДСТУ 3008-95 та стандартами ЄСПД	21.10.2019-04.11.2019	
4.	Проведення дослідження за темою БДР під керівництвом керівника	04.11.2019-30.12.2019	
5.	Завершення роботи над першим варінантом частини БДР	20.01.2020-03.02.2020	
6.	Проведення роботи експериментальною частиною БДР	03.02.2020-10.03.2020	
7.	Проведення роботи над програмним продуктом	10.03.2020-18.05.2020	
8.	Оформлення БДР та аналіз отриманих результатів	18.05.2020-07.06.2020	

Студент Антон Терещенко

Керівник Максим Древаль

* Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломної роботи.

РЕФЕРАТ

Дана диплома робота містить 140 ст. 4 ч., 8 табл., 19 рис., 2 дод., 18 джерел.

РОЗРОБКА РЕКЛАМНИХ СТРАТЕГІЙ, ІНТЕРНЕТ-МАРКЕТИНГ, МЕТОДИ ОПТИМІЗАЦІЇ

Об’єкт дослідження – показники рекламних стратегій у сфері інтернет-маркетингу.

Предмет дослідження – критерії розробки рекламної стратегії, а також методи оптимізації рекламних показників.

Мета та цілі роботи – розглянути методи та процес розробки стратегії рекламного просування, провести огляд існуючих інструментів рекламного кабінету соціальної мережі Facebook, розробка засобів програмного забезпечення для розробки стратегій рекламного просування у сфері інтернет-маркетингу.

Методи дослідження – вивчення алгоритмів майданчику та можливих варіантів інтерпретації результатів, спираючись на вхідні дані щодо існуючих тестів просування. Аналіз залежностей поміж стратегіями ставок на певні види оголошень та отримані результати залученості користувачів та їх охоплення.

Результатом роботи є автоматизована система розробки стратегії, а саме стратегії рекламного просування у сфері інтернет-маркетингу.

Новизною роботи є створення способу аналізу рекламних показників у рамках завершених рекламних кампаній, а також розробка програмного забезпечення з використанням знайденого способу.

Результати даної роботи можна застосовувати для полегшення та пришвидшення розробки стратегій рекламного просування с орієнтацією на показники часу та їх розподілення. Створений продукт є універсальним та може використовуватися із будь-якими рекламними кампаніями та продуктами, що просуваються.

ABSTRACT

This thesis contains 140 p., 4 sections, 8 tabl., 19 fig., 2 appendixes, 18 sources.

ADVERTISING STRATEGIES DEVELOPMENT, INTERNET MARKETING, OPTIMIZATIONS METHODS

The object of research – indicators of advertising strategies in the field of internet marketing.

The subject of research – criteria for developing an advertinsing strategies and optimization methods for it's advertising performance.

The purpose and aims of this work is to consider the methods of advertising promotion strategy development, to review the existing tools of the advertising office of the social network Facebook, to develop software for developing strategies for advertising promotion in the field of Internet marketing.

Research methods - study of site algorithms and possible options for results interpretation based on input data of existing promotion tests. Analysis of the relationships between bidding strategies for certain types of ads and the results of user engagement and coverage.

The result of the work is an automated system for strategy development, namely the strategy of advertising promotion in the field of Internet marketing.

The novelty of the work is the creation of a method of analysis of advertising indicators in the framework of completed advertising campaigns, as well as the development of software using the found method.

The results of this work can be used to facilitate and accelerate the development of advertising promotion strategies with a focus on time indicators and their distribution. The created product is universal and can be used with any advertising campaigns and promoted products.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

SMM – Social Media Marketing.

Таргетинг – рекламний механізм, що дозволяє виділити авдиторію, яка базується на певних критеріях.

Ретаргетинг – рекламний механізм, за допомогою якого оголошення направляються до тих користувачів, які вже бачили рекламу від певної компанії.

ERR – Engagement Reach Rate.

ОС – операційна система.

ЕОМ – електронно-обчислювальна машина.

ВСТУП

В наш час, час сучасних технологій та глибокої інтеграції соціальних мереж у життя людини, більшість компаній та підприємців малого бізнесу віддають перевагу рекламному просуванню на різних платформах, що забезпечує до їхніх мереж притік певного трафіку та реалізацію процесу лідогенерації.

У більшості випадків клієнт, що має намір рекламувати продукт, звертається до компаній вузького профілю, які працюють із рекламними інструментами найпопулярніших майданчиків. Найпоширенішим майданчиком є Facebook Business Manager, який дозволяє публікувати рекламні оголошення у Facebook та Instagram, а також забезпечує великий набір інструментів для аналізу трафіку, демографії, інтересів та авдиторій в цілому.

При розробці рекламної стратегії та запуску кампаній кожного разу проводяться контрольні тести, що призначені для аналізу залученості авдиторій різних вікових груп з певними інтересами на креатив (зміст рекламного оголошення).

Після тестів відбувається масштабування рекламних кампаній та починається робота у більш завантаженому режимі.

Для проведення тестів необхідно використовувати ресурс таргетолога, тобто людини, що займається рекламним просуванням у соціальних мережах за певними ознаками: стать, вік, інтереси, геолокація, мова спілкування, взаємодія із іншими рекламними оголошеннями, сімейне положення тощо.

При початку роботи створюється комерційна пропозиція, що відображає суть та її вартість. У подальшому ця робота коригується та відбувається процес тестування та аналізу наявних ресурсів клієнта: аудит сторінок, аудит рекламного кабінету, підхід до архітектури та структуризації контенту та рекламних оголошень в цілому.

Після проведення тестування та аналізу пріоритетна робота полягає у розробці стратегії рекламного просування у сфері інтернет-маркетингу, що включає в себе, але не обмежується:

а) приклади оформлення рекламних оголошень;
б) визначення цільової авдиторії;
в) аналіз показників рекламного кабінету;
    1) конверсія;
    2) охоплення;
    3) середнє органічне охоплення;
    4) середня кількість реакцій на публікацію;
    5) кількість підписників сторінки та її приріст;
    6) кількість залучених користувачів;
    7) середня кількість перегляду оголошень;
    8) ERR;
    9) загальне охоплення рекламних оголошень;
г) стратегія щодо подальшого просування за часом контакту з авдиторіями.

Тому об’єктом даної роботи є побудова допоміжної системи для розробки стратегії рекламного просування, яка буде працювати на сучасному персональному комп’ютері, локально та без необхідності підключення до інтернету.

Перший розділ більш детально розкриває актуальність задачі, обговорюються основні положення використання та інтерпретації рекламних алгоритмів Facebook, методи їх аналізу. У розділі 2 розглядається тема вибору методу реалізації цілі завдання, доступні варіанти оптимізації. Розділ 3 присвячений вибору конкретного методу оптимізації, його опису та реалізації, аналізу отриманих результатів. У розділі 4 буде проведений функціонально-вартісний аналіз роботи.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність задачі розробки рекламної стратегії

Соціальні мережі давно стали невід'ємною частиною абсолютної більшості населення планети. У них продають, інформують, запрошують на заходи, проводять розіграші, здійснюють опитування та навіть вибудовують репутацію компаній. Соціальні мережі знаходяться у режимі постійної битви за увагу користувача, якому пропонують все: від рукавів для випікання до вертольотів.

Тому виникає потреба не лише надавати матеріали, а й уважно відслідковувати реакцію користувачів на кожен вид контенту, аналізувати дії. Користувачі приходять у соціальні мережі не за рекламою, а за інформацією, яка може бути будь-якою. Наприклад завжди на очі може потрапити величезний текст про нафту. Якщо матеріал цікавий, користувач готовий його сприйняти.

Тим часом SMM постійно стикається зі зміною трендів. Лідери думок, що мають мільйони підписників, поступово відходять на другий план, а їх місце займають мікроінфлюесери, що мають приблизно 50-150 тисяч підписників. Вони мають більш чуйну та живу авдиторію, а їх послуги коштують значно дешевше. Тим паче авдиторія не так охоче реагує на поради зірок та блогерів, коли знає, що ці рекомендації оплачені брендами.

У рамках підвищення лояльності авдиторії бренди навіть запроваджують різноманітні системи комунікації, такі як чат-боти, що можуть пропонувати товари, допомагати орієнтуватися каталогом, давати відповіді на найпоширеніші запитання та створювати видимість розмови та піклування.

Найбільше цінується інтерактивний контент, адже людина, що заходить у соціальні мережі, вже звикла до схожих картинок, відеороликів, анекдотів та цитат. Вона потребує нових видів контенту. Таких, де їй відводиться не пасивна, а активна роль. Активне залучення іде на користь компанії, оскільки з'являється задоволення від конкретного отриманого контенту, асоціації та лояльність. Тим

паче гарна вікторина або головоломка можуть мати вірусний ефект, що зумовлює віральний ефект.

Насамперед, важливо вести стратегічне планування та мати комплексний підхід. Якщо ще декілька років тому було достатньо трьох кліків, щоб отримати потенційного клієнта, то вже зараз людина має отримати інформацію з п’яти різних джерел, щоб зацікавитись. Тому дуже важлива комплексна присутність та стратегія. Замість гонки за підписниками і лайками, необхідно підбирати релевантні сфері бізнесу майданчики та інструменти, які залучать увагу клієнта до покупки товару або послуги. Не обмежуючись лише соціальними мережами, можливо підключати різні канали просування, такі як сайт, email, мессенджери, контексту рекламу, блоги і т.д.

1.2 Історія розвитку реклами

Поняття рекламування відображалося ще за часів єгипетської, римської та вавилонських культур, та вражала темпами розвитку підходу до актуалізації пропозицій. Підприємці у особі торговців намагалися показати свій товар найбільш вигідно, описати усі його переваги, захопити увагу потенційного покупця, схилити його на свою сторону та укласти вигідну угоду.

Першим рекламним носієм можна вважати єгипетський папірус, який інформував про продаж рабів. Також вважається, що декі такі рекламні оголошення могли розміщатися на камінні та стінах. Один з таких каменів був знайдений у місті Мемфіс, який був першою столицею древнього Єгипту.

У Римі усі оголошення наносили фарбою на спеціальні дощечки, використання пергаменту почалося вже згодом. А на ринкових площах такі оголошення голосно зачитували, частіше за все роблячи це при великому скупченні народу.

Більш глобальні поштовхи відбувалися у момент появи першого друкарського верстату та, відповідно, появи книг. Розпочалася нова епоха засобів масової інформації, адже тепер з'являлась значна економія часу та можливість створювати матеріали у великих об'ємах. Вважається, що засновником друкованої реклами був французький лікар Теофраст Ренодо. У 1630 він відкрив у Парижі довідкову контору, яка займалася друком реклами у газеті «La Gazette». Першою рекламою заявлялась винагорода тому, хто знайде 12 вкрадених коней, пізніше таке оголошення опублікували і у лондонській газеті.

Перші торговці використовували рекламу не як засіб залучення авдиторій, а як систему оголошень, що певний товар є у наявності, наприклад сіль, борошно, хліб, сіно і т.д. Але згодом стало зрозуміло, що це не приносить достатнього попиту, тому у хід пішли різні засоби, за допомогою яких намагалися привернути увагу. Наприклад приказки, пісні, вірші, навіть невеличкі сюжети, що схиляли до уваги до деякого певного товару. У Парижі можна було зустріти навіть рекламу дешевих шинків у межах міста, де можна було випити відмінного вина.

У 1704 році у США з'явилася перша газета, яка була орієнтована виключно на рекламу – «Boston News-Letter». У ній публікувалися оголошення як від звичайних торговців, так і від фермерів. До того ж не завжди реклама саме пропонувала якийсь товару, траплялися випадки і пропозицій щодо роботи, покупки деяких добрив тощо. Розвиток реклами у США асоціювався із Бенджаміном Франкліним. Він створив «Пенсильванську газету», яка мала величезний тираж і велику кількість друкованої реклами.

Згодом, під час індустріальної революції в Англії, торговці зрозуміли справжню потужність та можливості реклами для отримання більш високого прибутку. Це призвело до зміни друкованої реклами, адже у 1839 році з'являється поняття фотографії. Тепер реклама могла мати вигляд не лише тексту, а й зображення певного товару, що викликало більшу довіру до продукту. А після винайдення у 1884 р. телеграфу реклама перетворюється в одну із

головних функцій маркетингу. Тепер реклама допомагала еліті створювати потреби для людини в умовах ринкової економіки.

З часом рекламою починають займатися спеціалізовані агентства та компанії, створювалися окремі підрозділи. Рекламні агентства починали свою роботу з купівлі ділянок землі та їх подальшого перепродажу із величезним прибутком. Рекламні агенти укладали угоди з газетами і журналами, а потім продавали право на розміщення іншим. У ті часи рекламодавці готували матеріали самостійно, а вже згодом цим почали займатися рекламі агентства. Перша агенція була створена у 1890 р. під назвою «Айер та син». Вони одні з перших планували та створювали рекламні оголошення.

У процесі розвитку рекламної справи та маркетингу в цілому ринкова економіка суттєво змінила зміст та форму реклами. Вона стала головною ланкою між споживачем та виробництвом.

Перша банерна реклама з’явилася в інтернеті на початку 1990 років. Одна з фірм масово розміщувала рекламу своїх юридичних послуг та цим привертала увагу. Згодом, із появою Facebook та запуском можливості рекламного просування почала з’являтися медійна реклама, яка передавала суть оголошення за допомогою текстів, логотипів, анімації, відео, фото, фотографій та інших графічних елементів. Рекламодавці, замість того щоб показувати рекламу всім, почали використовувати таргетинг. Таким чином вони відокремлювали користувачів за певними ознаками і посилювали вплив. З’являється поняття соціального медіа маркетингу – SMM. Цей підвид орієнтується на просування різноманітних матеріалів у соціальних мережах та медіа-майданчиках.

1.3 Реклама Facebook. Типи закупівлі та алгоритми аукціону

1.3.1 Реклама Facebook. Типи закупівлі

Тип закупівлі визначає способи оплати та таргетингу для рекламної кампанії. Facebook дозволяє обрати один з трьох типів:

а) Частота та охоплення

б) TRP

в) Аукціон

1.3.2 Тип закупівлі «Охоплення та частота»

Тип закупівлі «Охоплення та частота» (Reach and Frequency) надає можливість рекламодавцям планувати та купувати рекламну кампанію у рекламному кабінеті завчасно. Рекламні оголошення можна транслювати у Facebook, Instagram, Audience Network (для цілі «Перегляди відео»). Доступ до цього типу має обмежена кількість рекламодавців. Для роботи з цим типом закупівлі необхідно звертатися до служби підтримки соціальної мережі із окремим запитом. Вигляд рекламного кабінету зображено на рисунку 1.1.

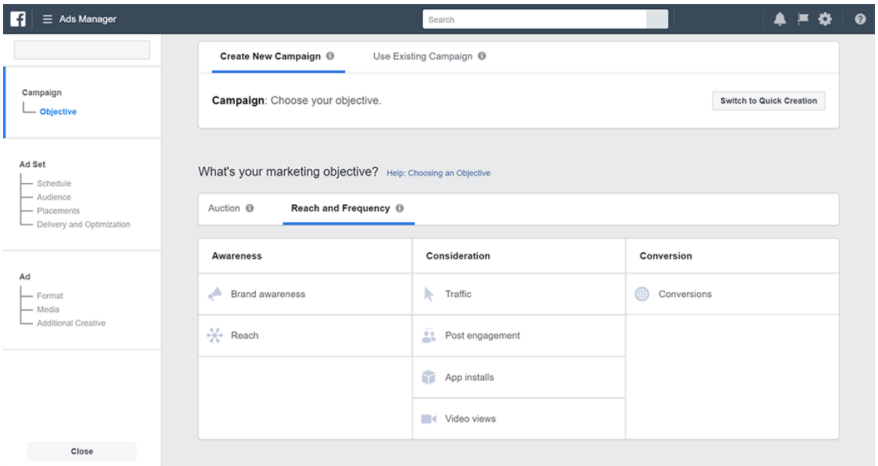


Рисунок 1.1 - Рекламний кабінет Facebook

1.3.3 Особливості роботи із типом закупівлі «Охоплення та частота»

а) Визначення точного бюджету, що необхідний для охоплення потрібної авдиторії, що дає рекламодавцю можливість контролювати витрати;
б) повний контроль показів рекламних оголошень. За допомогою цього типу можна з’являється можливість більш точно контролювати охоплення та частоту показів окремої кампанії. Інструмент дає можливість розуміти як часто користувачі будуть бачити оголошення ще до запуску рекламної кампанії;
в) послідовний показ. Дозволяє налаштувати порядок показу реклами на майданчику;
г) планування перед покупкою. Для вивчення усіх доступних можливостей і складання плану кампанії, можливо використовувати інструмент Campaign Planner. Цей інструмент допомагає створювати та порівнювати медіаплани, проте доступний він лише для певних типів закупівлі.

Випадки, у яких необхідно обрати тип закупівлі «Охоплення та частота»:

а) Охопити авдиторію більше ніж 200 000 чоловік;
б) Звернутися до цілої країни, а не до окремого регіону;
в) Отримати чітке прогнозоване охоплення своєї реклами;
г) Контролювати те, скільки разів користувачі будуть бачити певне рекламне оголошення;
д) Планувати та бронювати рекламні кампанії завчасно.

1.3.4 Тип закупівлі TRP (Target Rating Point)

Поняття TRP (Target Rating Point) походить з телевізійних метрик. Ця статистика мала назву GRP (Gross Rating Points) що рахується для певної цільової авдиторії, у порівнянні з усією можливою авдиторією.

Ця метрика була введена для оцінки ефективності відеооголошень.

Тип закупівлі TRP дозволяє рекламодавцям, які вже запускали телевізійні кампанії, планувати і купувати відеокампанії у Facebook та Instagram за допомогою підтвердженого Nielsen цільового рейтингу.

Nielsen Holdings PLC – незалежна глобальна кампанія, яка проводить маркетингові виміри в індустрії товарів повсякденного попиту. Компанія присутня більше ніж в 100 країнах світу, у яких проживають 90% населення Землі і які виробляють 90% глобального ВВП, надає клієнтам повноцінне бачення того, «що споживачі дивляться» (контент, рекламні ролики) і «що споживачі купують» (категорії, бренди, продукти), і допомагає їм виявити, як одне впливає на інше.

1.3.5 Особливості роботи з типом закупівлі TRP

а) TRP дозволяє замовляти рекламні кампанії за 6 місяців до їх показу
б) Цей тип закупівлі дозволяє також придбати певну кількість показів рекламної кампанії цільовій авдиторії за встановленою ціною
в) Оплачувати такі кампанії після того, як Nielsen покаже вашу рекламну кампанію цільовій авдиторії, демографічні данні якої ви дізналися.

Рекламну кампанію можна запустили із цим типом закупівлі лише за умови попереднього погодження із Facebook. Для цього необхідно чітко визначитися з наступними речами:

а) ціль TRP;

б) цільова авдиторія;

в) термін;

г) бюджет;

д) частота показів;

е) платформа: Facebook та Instagram.

1.3.6 Тип закупівлі «Аукціон»

До типу закупівлі «Аукціон» мають доступ усі рекламодавці. Він забезпечує більш широкий вибір параметрів, ефективність та гнучкість, але менш передбачувані результати. Використовуючи цей тип, можливо показувати рекламу на майданчиках Facebook та Instagram, Messenger і Audience network.

У аукціоні виграє та показується реклама із найвищою «сумарною цінністю» (рисунок 1.2).

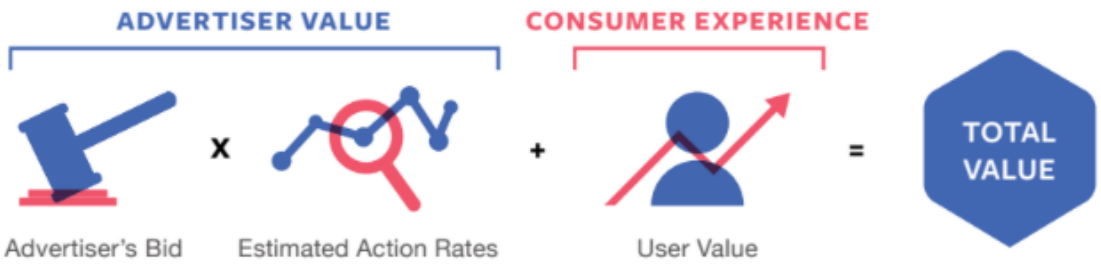


Рисунок 1.2 - Склад сумарної цінності

Сумарна цінність (Total Value) – це не та сума, яку повинен заплатити рекламодавець за показ своїх рекламних оголошень. Для визначення цього параметру враховуються такі фактори:

а) ставка;

б) приблизна частота дій;

в) якість та актуальність реклами.

1.3.7 Ставка

Ставка – це число, що відображає те, скільки ви готові витратити. Також це інструмент для контролю ціни у випадку перемоги на аукціоні.

Існує два види ставок:

Мінімальна ціна

Ставка встановлюється алгоритмом Facebook. Якщо встановлюється межа ставки, то з’являється можливість керувати ціною за подію оптимізації.Але якщо ставка при цьому буде занадто низька, кількість показів суттєво зменшується.

Цільова ціна

Ставка встановлюється рекламодавцем. Ця стратегія дозволяє досягти під час показу рекламних оголошень середньої ціни за подію оптимізації, яка буде як можна ближче до встановленої цільової ціни.

Вікна вибору максимальної ставки та вибору вікна конверсії зображені на рисунку 1.3 та рисунку 1.4 відповідно.

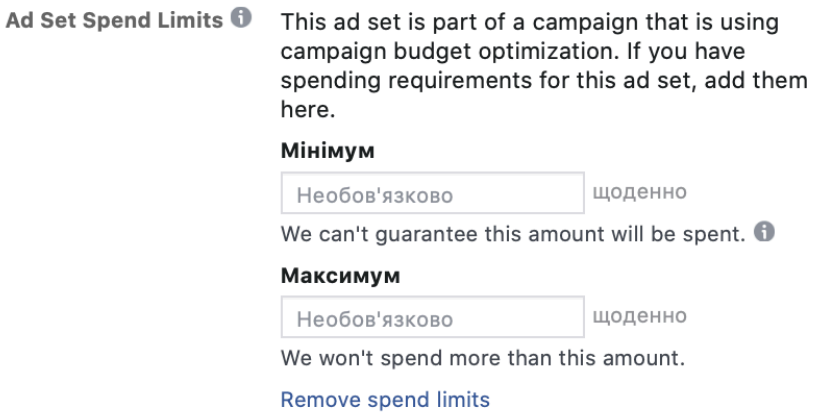


Рисунок 1.3 - Вікно вибору максимальної та мінімальної ставки

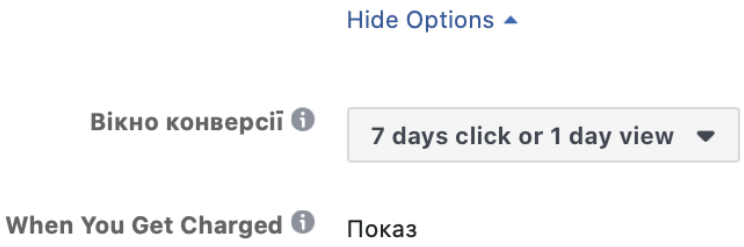


Рисунок 1.4 - Вікно вибору вікна конверсії, аркуш 20

1.3.8 Приблизна частота дій

Приблизна частота дій – це показник, який вказує на вірогідність здійснення користувачем цільової дії: клік, репост, скачування мобільного додатку, коментарю, лайку і т.д.

Якщо цей показник низький, сумарна цінність реклами на аукціоні буде вище. У такому випадку Facebook підвищує ставку, щоб компенсувати низьку приблизну частоту дій. Але це відбудеться лише у тому випадку, якщо обрана автоматична ставка.

1.3.9 Якість та актуальність реклами

Для кожного показу реклами Facebook обирає найкращі оголошення з врахуванням не тільки вищевказаних показників, а й результативності реклами.

Більшість ресурсів вказує, що необхідно звертати увагу на метрику Relevance Score – коефіцієнт релевантності. Але тут є певні зауваження від платформи.

Згідно заяв, коли реклама потрапляє на аукціон, значення Relevance Score не використовується при розрахунку загальної цінності, яка визначає переможця.

По цій причині (а також тому, що коефіцієнт є відносним значенням) платформа рекомендує не приділяти занадто багато уваги підвищенню коефіцієнту релевантності. Рекомендовано використовувати коефіцієнт, щоб отримати загальне уявлення про актуальність конкретної реклами і зосередитися на покращенні таргетингу і оформленню.

При цьому дуже важлива історія рекламної кампанії. Низький інтерес до рекламної кампанії або негативні реакції від користувачів будуть мати величезний вплив – Facebook підвищує ціну показу рекламних оголошень. Тому важливо, щоб рекламні оголошення були спрямовані на близькі до бренду авдиторії.

1.3.10 Особливості конкуренції на ринку аукціонів

В певні дні та години рівень конкуренції на аукціоні може збільшуватися. Це значить, що більшість рекламодавців намагається охопити одну й ту ж саму авдиторію. У такі моменти швидкість витрат бюджету може зменшуватися, або ж мінятиметься ціна за результат.

Якщо група оголошень активна щонайменше 5 днів і не має 500 показів, стає доступним Delivery Insights – панель із різними метриками, що допомагають зрозуміти ефективність результатів рекламних груп оголошень.

Delivery Insights пропонує рекламодавцям наступні метрики для відслідковування впливу конкуренції на роботу груп оголошень:

а) Ставка, встановлена початково для певної групи оголошень. Результативність рекламної кампанії в більшості залежить від обраної ставки.

б) Сума ставки, вказана системою на початку аукціону (згідно до темпу системи показу Facebook) – рівномірна аукціонна ставка. Наприклад, якщо при створенні груп оголошень ставка встановлюється вручну, то

рівномірна аукціонна ставка може бути іншою. Причина в тому, що ставка системи показу реклами може бути менше встановленої. Це відбувається для того, щоб рівномірно розподілити бюджет у період дії кампанії. Якщо рівномірна аукціонна ставка приблизно рівна до встановленої, це значить, що бюджет витрачається недостатньо швидко.

в) Зміна конкуренції на аукціоні. Дана метрика показує наскільки змінилась конкуренція в певний обраний день по відношенню до конкурентноздібної рівномірної аукціонної ставки.

Принцип роботи алгоритму Facebook базується на принципі роботи аукціону другої ціни. Виграє той, хто запропонував максимальну ставку. Але переможець платить другу вказану ціну +1 цент.

1.4 Інструмент Facebook Analytics

Facebook Analytics допомагає виміряти і зрозуміти активність користувачів на вашому сайті, в додатку, в боті в Facebook Messenger, в Facebook або Instagram, а також в іграх з Facebook Gameroom. Якщо вже є доступ в Business Manager, додатково підключати нічого не потрібно.

Facebook Analytics не замінить Google Analytics або інші подібні інструменти, але допоможе отримати більше даних про поведінку користувачів.

1.4.1 Навіщо потрібен Facebook Analytics

а) Люди заходять в Facebook або Instagram під одним аккаунтом як з телефону, так і з комп'ютера, тому аналітика виходить people-based, а не cookie-based.

б) Можна користуватися різними корисними звітами: будувати воронки, вивчати шлях користувача на їх основі і бачити, де є проблемні місця. Можна будувати когорти, де наочно буде видно весь LTV користувача.

в) Дані можна отримувати і візуалізувати для різних сегментів за допомогою вбудованих фільтрів.

г) Можна групувати джерела даних і канали, щоб краще розуміти, як поводяться користувачі. Наприклад, об'єднавши Instagram і піксель Facebook, можна побачити, чи роблять ваші передплатники покупки на сайті.

д) Можна створювати кастомні авдиторії, застосовуючи різні фільтри прямо з інтерфейсу Facebook Analytics, щоб потім використовувати їх для ретаргетингу і побудови look-a-like-аудиторій.

На відміну від Facebook Ads, в Analytics збираються дані як по платних каналах, так і по органічним, як зображено на рисунку 1.5.

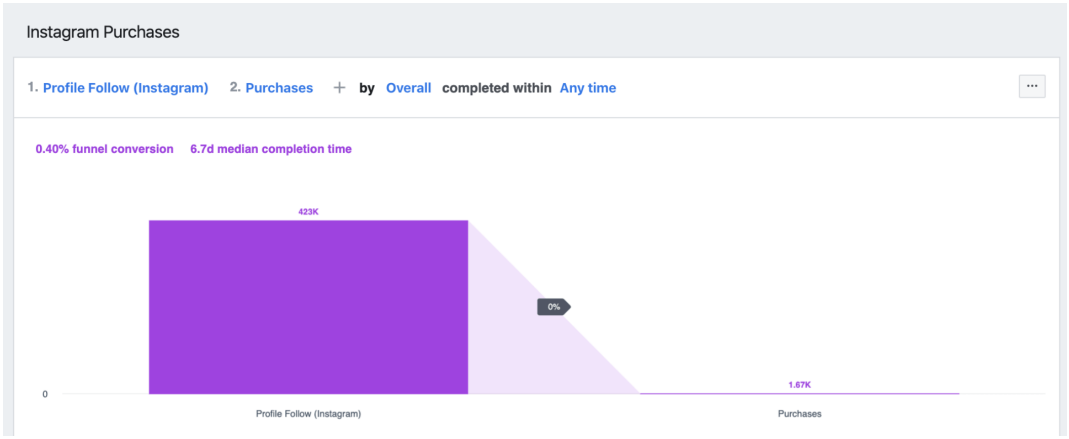


Рисунок 1.5 - Конверсія підписників в Instagram на покупку на сайті (0,4%)

1.4.2 Основні поняття

Перш ніж почати користуватися інструментом, потрібно визначитися з основними поняттями, якими оперує Facebook.

а) Події - інформація про дії користувача, яку зміг зібрати Facebook. Події бувають як стандартні (AddToCart, Purchase), так і призначені для користувача (кастомні).

б) Параметри - додаткова інформація про події (наприклад, вартість товару). Вони також можуть бути стандартними і призначеними для користувача.

в) Джерела даних - місця, звідки Facebook отримує дані про події.

г) Канали - спосіб взаємодії з продуктом і середовище, що використовується для реєстрації подій.

Facebook спеціально зробила тестовий аккаунт, щоб можна було самостійно розібратися в інструменті.

Також є окремий довідковий центр і блог, де можна знайти відповіді на основні питання і дізнатися про останні оновлення.

Після того як додаються джерела і канали з даними, користувач потрапить на сторінку Overview. Тут можна подивитися графіки з найважливішими даними за версією Facebook. Поруч з кожним графіком є кнопка, клікнувши по якій, можна побачити повний звіт за цими даними.

1.4.3 Фільтри

За допомогою фільтрів можна отримувати дані сегментів, що вас цікавлять. Залежно від обраного звіту дані можна фільтрувати:

а) За подіями

б) За демографічними даними (вік, стать, країна і мова)

в) За інформацією про пристрої (наприклад, модель пристрою, браузер або версія програми)

г) За джерелами встановлення програми

д) За веб-параметрам (джерелами трафіку, UTM-мітках, доменів, URL)

е) Крім того, з усіма фільтрами можна працювати за допомогою операторів «is», «is not», «contains», «does not contain» та інших, щоб отримати потрібний сегмент

У деяких звітах («Активні користувачі», «Виручка», «Події») є можливість порівнювати два набори даних (рисунок 1.6), використовуючи фільтр порівняння. Наприклад, можна порівняти доходи, які принесли користувачі Android і iOS.

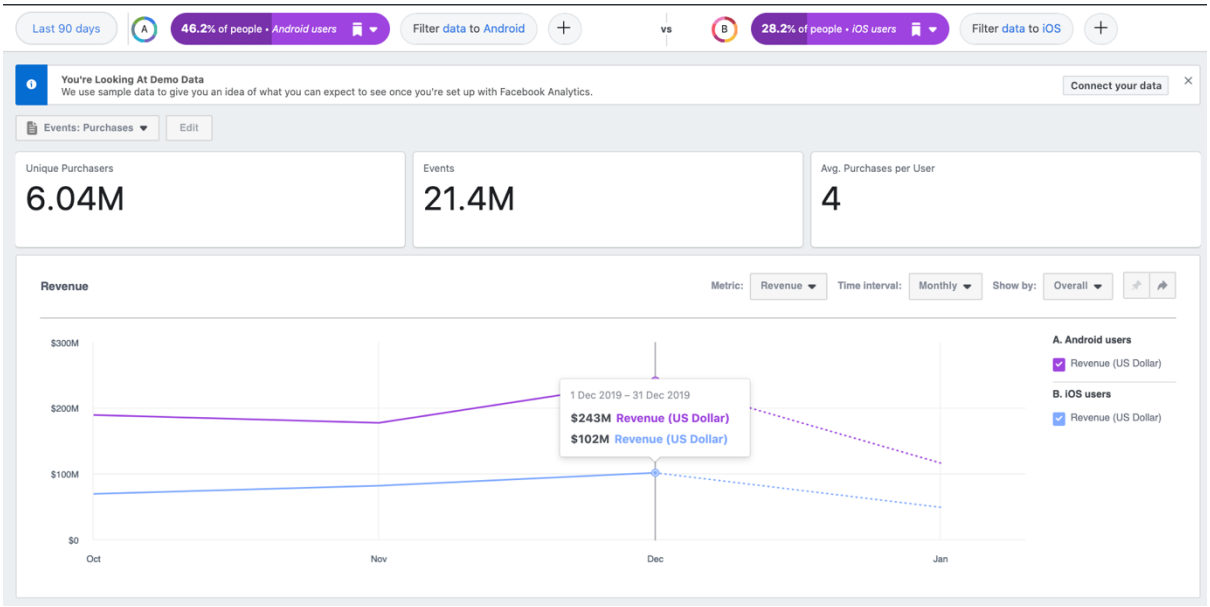


Рисунок 1.6 - Вікно порівняння наборів даних

Можна фільтрувати сегменти за процентним співвідношенням: наприклад, можна виділити сегмент, який купує частіше, ніж 83% вашої аудиторії, і створити з нього модифіковану авдиторію для ретаргетингу.

Фільтри можна зберігати і ділитися ними, щоб інші користувачі теж могли бачити і аналізувати отримані сегменти.

1.4.4 Активні користувачі

З цього звіту можна дізнатися, скільки людей взаємодіяли з вашим додатком, пікселем або групою джерел подій. Залежно від джерела подій відображає такі дані:

а) Дії користувачів: кількість подій, унікальне кількість користувачів, липкість (кількість активних користувачів за день, поділене на кількість активних користувачів за місяць), середня кількість подій на користувача і MAU / WAU / DAU.

б) Сесії: середня тривалість сеансу, медіанна тривалість сеансу, кількість сеансів.

в) Вік і стать.

1.4.5 Виручка

Тут можна побачити всю інформацію про покупки ваших користувачів (рисунок 1.7):

а) Вартість покупок;

б) дані за подією покупки;

в) вік і стать покупців.

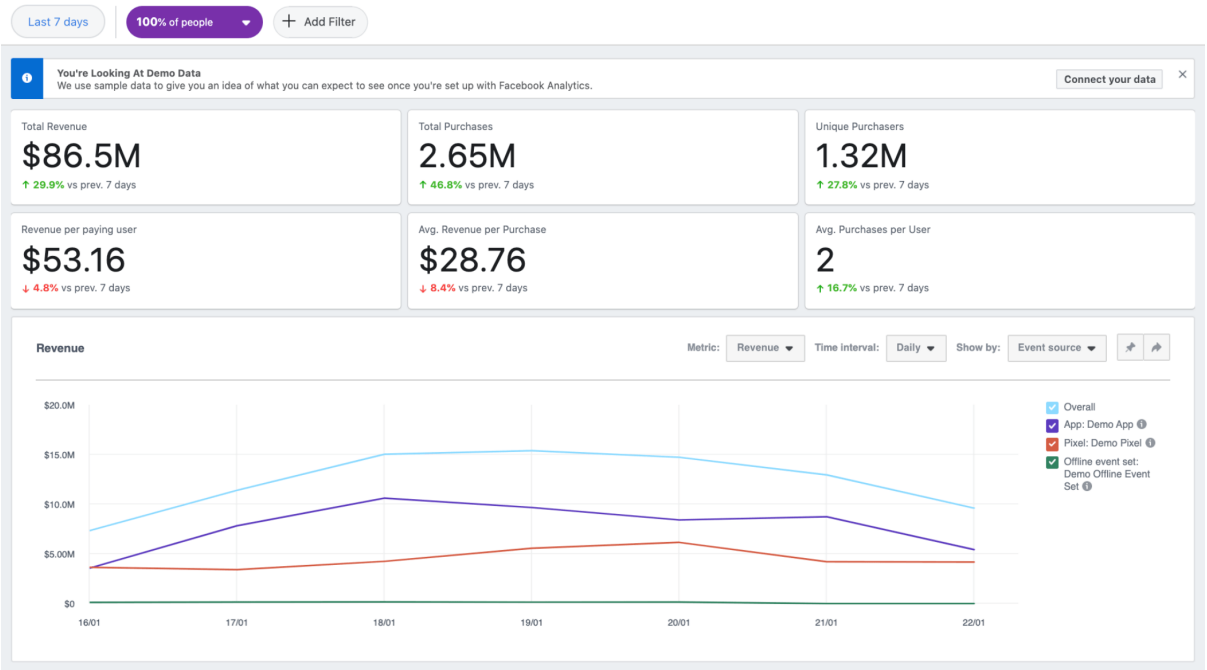


Рисунок 1.7 - Узагальнені дані про прибутки з декількох джерел даних, аркуш 27

1.4.6 Воронки

Один з найбільш корисних звітів. Він допомагає зрозуміти послідовність дій користувачів перед покупкою, за відгуками друзів кожного етапу воронки і час, який потрібно користувачам для переходу на наступний етап.

Тут же можна побачити, де, коли і скільки користувачів відвалюється і на основі цих даних зрозуміти, на якому етапі потрібно щось оптимізувати.

1.4.7 Утримання

Звіт показує відсоток людей, які повертаються в ваш канал (сторінка на Facebook, iOS-додаток) після першої дії, яким може бути, наприклад, перший перегляд сайту або встановлення додатку.

1.4.8 Когорти

Цей звіт допомагає вивчити групи людей, які скоїли дві обраних вами дії протягом заданого проміжку часу. Залежно від обраних подій і параметрів когорти допомагають отримати дані про утримання клієнтів, цінності життєвого циклу або частоту повторних покупок.

Дані, отримані з звітів по воронок і когортам, можуть бути аналогічними, якщо вам потрібно тільки виміряти якусь послідовність дій.

Однак в когортах відображаються дані згідно заданої розбивки (по днях, тижнях, місяцях), тобто ви, наприклад, зможете побачити, як вплинула ваша додаткова маркетингова кампанія на підсумкову конверсію в рамках місяця, якщо вона була запущена в другий тиждень місяця і тривала тиждень (рисунок 1.8).

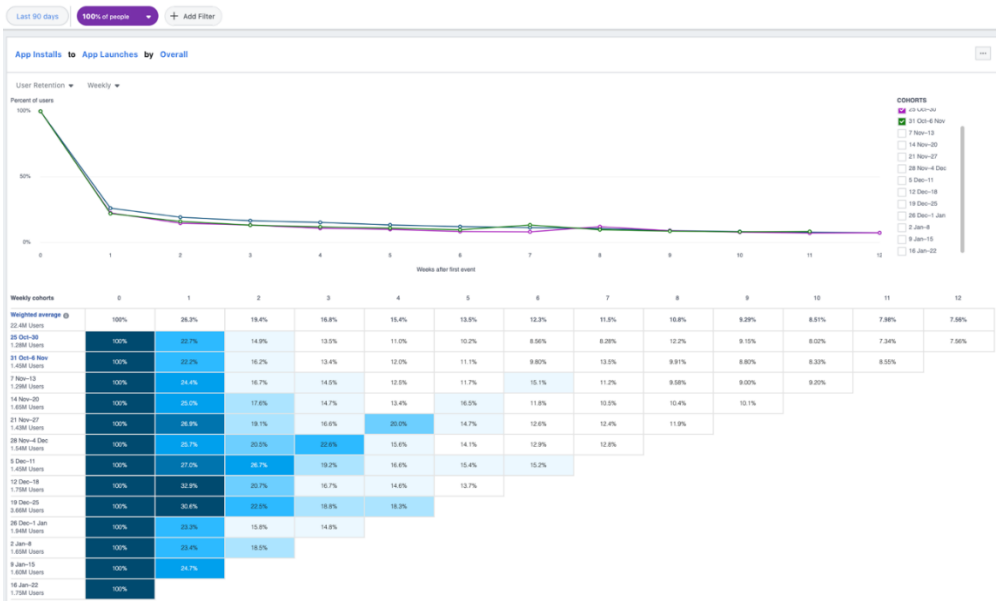


Рисунок 1.8 - Когортний аналіз у Facebook Analytics

1.4.9 Демографічні дані та інтереси

У цьому звіті представлена інформація про стать, вік, країну, місто та мову користувачів. У бета-версії звіту також доступні сімейний статус, рівень освіти і посада людей з обраного сегмента.

У розділі «Інтереси» відображаються категорії сторінок, на які підписані ваші користувачі, і самі сторінки, найбільш релевантні для вашої аудиторії. Ці відомості разом з інформацією з розділу «Демографічні дані» можуть дати додаткові інсайти для налаштування і оптимізації рекламних кампаній.

1.5 Визначення авдиторій та спліт-тестування. Оптимізація витрат на A/B тестування

1.5.1 Базові принципи

Можна виділити основні базові принципи A/B-тестування (які підходять для будь-яких схожих каналів просування).

а) На кожну авдиторію своя підкампанія з однаковими креативами - в ідеалі кампанію потрібно ділити на складові за статтю, віком, пристроями і географією.

б) На кожну аудиторію розрахувати окремий бюджет і максимальну ставку (maximal bid), оскільки у неї є свій набір показників, найчастіше однорідний, відповідно і результат кампанії для неї буде відрізняється.

в) Кожній аудиторії показати один і той же набір креативів, але з деякими відмінностями. Десятки тисяч загальнодоступних кейсів показують, що різні аудиторії по-різному сприймають інформацію і реагують на неї. Таким чином можна з'ясувати, який набір і яка комбінація покаже кращий результат.

Бюджет на авдиторію залежить як від її охоплення, CTR і CR, так і від очікуваної ціни за одиницю результату. Наприклад, якщо авдиторія складається з 10 тисяч чоловік, очікувані (передбачені на досвіді інших кампаній) CTR = 2%, CR = 20%, а очікувана ціна за одиницю результату не дорожче $5, то бюджет повинен бути близько 10 000 * 2 % * 20% * $5 = $200.

Максимальна ставка - це максимальне значення грошей, які рекламодавець готовий віддати за одиницю реклами або за одиницю результату на аукціонному майданчику. Вона залежить від типу кампанії і її цілей. Наприклад, для широкої авдиторії варто встановити максимальну ставку за вартість кліка, яка дорівнює значенню KPI щодо кліка; а при вузькій авдиторії зазвичай фахівці рекомендують її брати вище, ніж KPI (незважаючи на ризик надвитрат збільшується ймовірність показу саме вашої реклами серед всіх конкурентних оголошень для кожної людини даної аудиторії, а значить, і ймовірності збільшення кількості показів, що приносять потрібні кліки та конверсії) .

По закінченню кожної ітерації тесту можна ускладнювати гіпотези і перевіряти їх. Але для аукціонних платформ, подібних Facebook, існують обмеження щодо охоплення та бюджету.

1.5.2 Інструмент Facebook Auto Splitting

Незважаючи на те, що базові принципи оптимізації досить прості, вони вимагають великої кількості часу з боку менеджера. Дії складаються з наступних пунктів:

а) Заклад великої кількості підкампаній (адсетів).
б) На кожну підкампанію (адсет) генерація комбінацій креативів (при 2 картинках, 2 текстах і 2 заголовках виходить 2 * 2 * 2 = 8 комбінацій)
в) Постійне відстеження результатів (в ідеалі кожну годину).

г) Можна скористатися автоматизацією кроків 1 і 2 за допомогою реалізації розбиття таргетингу у вигляді підкампаній.

Алгоритм роботи:

а) Для будь-якого типу кампанії задаються базові налаштування (мета, формат, назва, бюджет і так далі).
б) На етапі деталізації таргетингу вказується основна авдиторія, тобто її гео, стать, вік, місце проживання (наприклад, Україна, 18-50, стрічка новин на робочому столі і в мобільних пристроях).
в) Переключається з ручного режиму в режим Auto Splitting.
г) Обирається, по якому полю будуть ділитися кампанії (зараз доступно розбиття за віком, статтю та пристроями, а так само з будь-якої комбінації цих параметрів).
д) Обирається метод розподілу бюджету (evenly поділить бюджет рівними частинами, by reach пропорційно ділить бюджет на основі охоплення конкретної аудиторії, наприклад «Україна, М 18-25» отримає більше бюджету так як він більше за обсягом, ніж «Україна, М 65+») .
е) Завантажуються креативи, заповнюються поля з повідомленнями і відбувається запуск кампанії.
ж) Очікування 2-3 хвилини (процес поділу може зайняти тривалий час) і отримання кампанії, в якій багато підкампаній (адсетів) відповідно до раніше обраного розбиття.

1.6 Висновки

У даному розділі було розглянуто актуальність задачі розробки рекламної стратегії та інструменти, що дозволяють аналізувати статистику рекламного просування за декількома параметрами. Також було приділено увагу історії

розвитку рекламної справи та основним тенденціям у сфері інтернет-маркетингу. Очевидно, що задача має високий потенціал, сьогодні у сфері діяльності цієї області активно ведуться розробки провідними агенціями світу. Було проаналізовано типи закупівлі рекламних оголошень та алгоритми відображення адсетів.

Досліджено переваги та недоліки використання різних типів закупівлі та інструментів аналізу даних усередині платформи. Також розглянуто декілька основних проблем, з якими сьогодні зіштовхуються системи аналізу рекламних авдиторій та, зокрема, рекламодавці.

2 ПІДХОДИ ДО ОПТИМІЗАЦІЇ ПОКАЗНИКІВ

2.1 Проблематика

Більшість задач, які виникають і різних сферах нашого життя, та у яких необхідно знайти оптимальне рішення, можуть бути сформульовані як задачі оптимізації. Складність об’єктів, які необхідно оптимізувати, постійно зростає, що так само стрімко призводить до ускладнення їх математичних моделей. Ця проблема значно ускладнює пошук оптимальних параметрів та знеможливлює знаходження оптимальної комбінації аналітично – з’являється потреба побудови числових методів для її пошуку.

Проблема числового пошуку вирішення задачі оптимізації, в свою чергу, може бути пов’язана із значними труднощами. У більшості проблема пов’язана із розмірністю та видом оптимізуємої цільової функції. У такому випадку необхідно орієнтуватися на задачу із такими властивостями:

а) багатоекстремальна
б) недефиренційована
в) задана у формі чорної скриньки – у вигляді деякої обчислювальної процедури або прибору, на вхід якого надходить аргумент, а на виході відображене відповідне значення функції.

2.2 Методи глобальної оптимізації

У рамках пошуку підходу до вирішення задачі необхідно звернути увагу на методи глобальної оптимізації. Ці методи суттєво відрізняються від методів локального пошуку, які є більш стандартними та часто нездатні знайти глобальний, тобто абсолютно кращий, розв’язок розглянутих задач через багатоекстремальну функцію.

Як правило виявляється, що локальні методи не здатні покинути зону притяжіння локальних оптимумів і, відповідно, втрачають глобальний оптимум, а використання знайдених локальних розв’язків може виявитися недостатнім, бо глобальний розв’язок може дати більш значний виграш у порівнянні з локальним.

Через високу складність таких задач їх не можуть ефективно вирішувати і прості методи перебору. Можливість побудови адаптивних систем, відмінних від переборних, для швидкого пошуку глобального розв’язку багато екстремальних задач пов’язана з наявністю і обліком апріорних припущень про характер проблеми, що розглядається. Такі припущення грають ключову роль при побудуванні швидких алгоритмів глобального пошуку і служать основним математичним інструментом для отримання оцінок глобального рішення після зупинки алгоритму.

Для багатьох практичних задач (таких як, наприклад, вирішення систем нелінійних рівнянь і нерівностей; оптимізація ієрархічних моделей, пов'язаних з завданнями розміщення, системами обслуговування і т. д.) типовим є припущення про Ліпшицевість функцій, що характеризують досліджувану систему . Це пов'язано з тим, що відношення збільшення функцій до відповідних збільшень аргументів зазвичай не можуть перевищувати деякого порогу, що визначається обмеженою енергією змін в системі, який може бути описаний з використанням константи Ліпшица. Розробкою теорії і методів чисельного рішення задач подібного типу займається ліпшицева глобальна оптимізація. Важливість даної підобласті глобальної оптимізації пояснюється як наявністю великого числа прикладних задач, що моделюються за допомогою ліпшицевих функцій, так і просторістю класу таких функцій.

2.2.1 Ліпшицева глобальна оптимізація

Можливість побудови адаптивних схем, відмінних від переборних, для швидкого пошуку глобального вирішення багатоекстремального завдання пов'язана з наявністю і урахуванням деяких апріорних припущень про характер даної проблеми.

Такі припущення грають ключову роль при розробці швидких алгоритмів глобального пошуку і служать основним математичним інструментом для отримання оцінок глобального рішення після зупинки алгоритму. При відсутності припущень про цільової функції і обмежень будь-яке як завгодно велику кількість обчислень цільової функції не дає гарантії знаходження глобального мінімуму, так як її поведінка може відрізнятися дуже вузькими і глибокими піками і западинами, які можуть перебувати між точками обчислень.

У рамках глобальної оптимізації розглядаються різні припущення про структуру цільової функції і обмежень (як, наприклад, безперервність, лінійність, опуклість, диференційованість і т.д.). Одним з природних і важливих (як в теоретичному, так і в прикладному відношеннях) припущень про завдання є припущення про обмеженість відносних змін цільової функції f (x) і обмежень gi (x), $1 \le i \le m$. Воно пов’язане з тим, що відносини збільшень функцій до відповідних збільшень аргументів зазвичай не можуть перевищувати певного порогу, що визначається обмеженою енергією змін в системі, який може бути описаний з використанням позитивної константи. У такому випадку функції називаються ліпшицевими, а завдання - завданням ліпшицевої глобальної оптимізації.

Формально дійсна функція ψ (x), визначена на множині $X \subseteq RN$, є ліпшицевою, якщо вона задовольняє умові (званій умові Ліпшиця):

$$\left|\psi(x') - \psi(x'')\right| \le L \left|\left|x' - x''\right|\right|, \quad x', x'' \epsilon X, \tag{2.1}$$

де $L = L(\psi, X)$ – деяка константа (константа Ліпшиця), $0 < L < \infty$,

||-|| - норма у просторі $\mathbb{R}^N$.

Умова Ліпшиця допускає ясну геометричну інтерпретацію. Припустимо, що одномірна ліпшицева функція ψ (x) (з відомою константою Ліпшиця L) була обчислена в двох точках, x' і x" (рисунок 2.1).

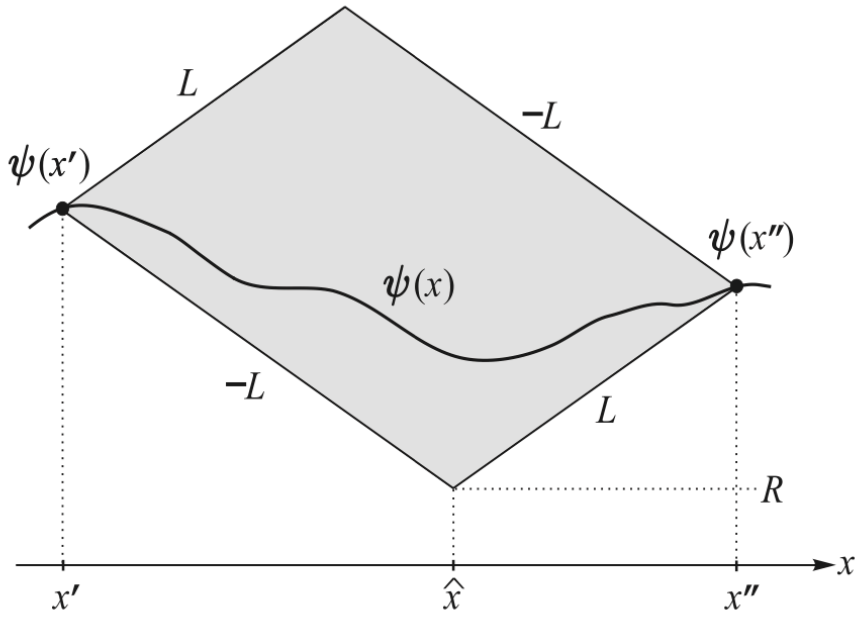


Рисунок 2.1 - Графік функції ψ (x), що задовольняє умові Ліпшиця на інтервалі [x', x"]

В силу умови на інтервалі [x', x"] (тут і далі при використанні терміну інтервал мається на увазі замкнутий безліч точок) виконуються наступні чотири нерівності (2.2)-(2.5):

$$\psi(x) \le \psi(x') + \mathrm{L}(x - x'), \quad x \ge x', \tag{2.2}$$

$$\psi(x) \ge \psi(x') - \mathrm{L}(x - x'), \quad x \ge x', \tag{2.3}$$

$$\psi(x) \le \psi(x'') - \mathrm{L}(x - x'), \quad x \le x', \tag{2.4}$$

$$\psi(x) \ge \psi(x'') + \mathrm{L}(x - x'), \quad x \le x'. \tag{2.5}$$

Таким чином, графік функції на інтервалі [x', x''] повинен розташовуватися всередині області, обмеженої чотирма лініями, що проходять через точки (x', ψ (x')) і (x'', ψ (x'')) з кутовими коефіцієнтами L і -L (світло-сірий паралелограм на рисунку 2.2.1).

Нескладно бачити, що може бути отримана оцінка найменшого значення функції ψ (x) при її мінімізації на інтервалі [x', x'']. З огляду на умови Ліпшиця (2.1) виконується нерівність (2.6):

$$\psi(x) \geq \overline{\psi}(x). \qquad (2.6)$$

де $\overline{\psi}(x)$ – кусково-лінійна функція (2.7) на інтервалі [x', x''],

$$\overline{\psi}(x) = \max\{\psi(x') - L(x - x'), \psi(x) + L(x - x)\}, x \in [x', x'']. \qquad (2.7)$$

Умова Ліпшиця може бути використана для оцінки глобального мінімуму функції на інтервалі, а знання константи Ліпшиця дозволяє конструювати алгоритми глобального пошуку і доводити умови їх збіжності.

2.2.2 Способи оцінки константи Ліпшиця

Найчастіше розглядається безліч різних алгоритмів розв'язання задачі. Вони можуть бути розділені як мінімум на чотири групи в залежності від способу отримання оцінки константи Ліпшиця.

До першої групи належать алгоритми, в яких застосовується деяка задана апріорно оцінка константи L. Такі алгоритми важливі з теоретичної точки зору, але їх застосування на практиці обмежене, так як при зазначених припущеннях про цільові функції завжди апріорна інформація про константу Ліпшиця часто недоступна.

Друга група включає алгоритми, що адаптивно оцінюють в ході пошуку глобальної константи Ліпшиця в усій області D, що є набагато більш практичним підходом. Однак оцінка константи Ліпшиця, що отримується при цьому, здатна надати лише слабку інформацію про поведінку цільової функції в кожній конкретній підобласті області пошуку. Алгоритми з використанням адаптивного оцінювання локальних констант Ліпшиця Li в різних зонах Di ⊆ D області пошуку D, що формують третю групу, дозволяють налаштуватися на локальну поведінку цільової функції у різних частинах допустимої області.

Нарешті, до четвертої групи належать алгоритми, в яких оцінки константи Ліпшиця вибираються з деякої множини можливих значень.

Відзначимо, що константа Ліпшиця істотно впливає на швидкість збіжності алгоритмів ліпшицевой глобальної оптимізації, тому настільки важливим є питання її коректної оцінки. Занижена оцінка істинної константи Ліпшиця L може привести до втрати глобального рішення. У той же час занадто велике значення оцінки константи L для мінімізуємої цільової функції передбачає складну структуру функції з різкими перепадами її значень і вузькими зонами тяжіння точок мінімумів. Тому завищена оцінка константи L, яка не відповідає істинному поведінки функції, тягне за собою повільну збіжність алгоритму до точки глобального мінімуму.

На рисунку 2.2 наведено приклад одновимірної функції з великим значенням глобальної константи Ліпшиця L на інтервалі [a, b]. Точка глобального мінімуму x* знаходиться в широкому інтервалі [a, c], де локальна константа Ліпшиця мала.

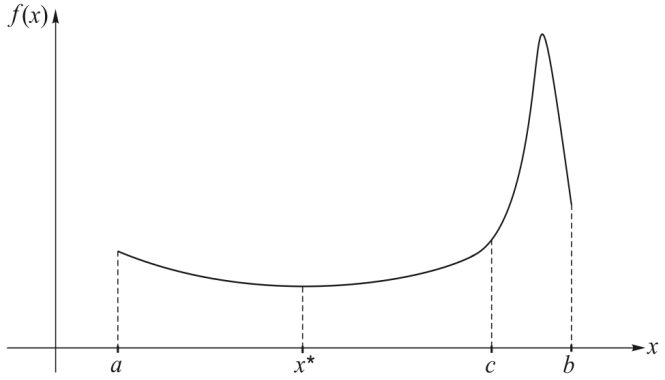


Рисунок 2.2 - Приклад функції, при мінімізації якої методи, які використовують під час пошуку лише глобальну константу L (або про її оцінку), будуть працювати повільно, аркуш 39.

Глобальна ж константа Ліпшиця збігається з локальної константою інтервалу [c, b], який дуже вузьке. У цій ситуації методи, які використовують під час пошуку лише глобальну константу L (або її оцінку), будуть працювати повільно на інтервалі [a, c], незважаючи на простоту функції в цій підгалузі і трохи локальної константи Ліпшиця, оскільки глобальна константа Ліпшиця велика.

Таким чином, поведінка методу в околиці x* буде залежати від локальної константи інтервалу [c, b] (і глобальної для всієї області визначення [a, b]), який не тільки малий і знаходиться далеко від точки x*, але ще і містить точку глобального максимуму. Тому використання тільки глобальної інформації про поведінку цільової функції в ході її мінімізації може значно уповільнити збіжність алгоритму до точки глобального мінімуму.

Традиційним прийомом подолання цієї проблеми є зупинка методу глобального пошуку з подальшим підключенням якогось локального алгоритму, який покращує отримане рішення на заключній фазі пошуку. При цьому підході виникають деякі складнощі, пов'язані зі сполученням глобальної та локальної процедур. Однією з них є питання про визначення моменту зупинки глобального алгоритму: передчасна зупинка може призвести до втрати глобального рішення,

пізня - уповільнює пошук. Інша проблема цього підходу полягає в тому, що локальна інформація про поведінку цільової функції використовується тільки в околиці глобального мінімуму. При цьому цілком очевидно, що можуть існувати зони області визначення, які знаходяться далеко від точки глобального мінімуму, але локальні константи Ліпшиця в яких істотно менше глобальної.

2.3 Безумовна оптимізація

2.3.1 Числові методи першого порядку. Градієнтний метод та його варіації

Одним із класичних методів оптимізації першого порядку можна назвати градієнтний метод. При його використанні мається деяка функція *f*, яка замінюється в околі чергової точки x^k її лінійною частиною розкладу в ряді Тейлора.

У градієнтному методі послідовність наближень змінної x до точки мінімуму буде мати вигляд (2.8):

$$x^{k+1} = x^k - \alpha_k f'(x^k), \quad \alpha_k > 0, \quad k = 0,1,2,\dots \tag{2.8}$$

При цьому необхідно обрати довжину кроку. У даному випадку будемо базуватися на умові мінімізації функції уздовж напряму антиградієнта. Таким чином ми отримуємо варіант найскорішого градієнтного спуску. Також існують практики, коли використовують інші методи, наприклад дроблення, де методи пошуку оптимального значення довжини кроку є наближеними.

Цільова функція у такому випадку має додаткові умови, наприклад опуклість. Якщо функція не опукла, градієнтний метод забезпечуватиме лише збіжність до множини стаціонарних точок *f*. Використаємо теореми про збіжність градієнтного методу.

Теорема 2.1 Нехай функція f диференційована та обмежена знизу на $\mathbb{R}^N$, а її градієнт задовольняє умову Ліпшиця. Тоді за довільної початкової точки $x^0 \in \mathbb{R}^N$ для методу (2.8).

Теорема 2.2 Нехай функція f двічі диференційована та сильно опукла на $\mathbb{R}^N$, а її матриця других похідних задовольняє умову (2.9)

$$(f''(x)h, h) \leq D\left||h^2|\right|, \qquad x, h \in \mathbb{R}^n \tag{2.9}$$

де h – вектор, що задає напрям спадання функції f.

Тоді за довільної точки $x^0 \in \mathbb{R}^N$ послідовність x^k $(k \geq 1)$, визначається формулою (2.8), збігається до точки мінімуму x^* із швидкістю геометричної прогресії (2.10)-(2.11):

$$f(x^k) - f(x^*) \leq q^k\left(f(x^0) - f(x^*)\right), \tag{2.10}$$

$$\left||x^k - x^*|\right| \leq C. \tag{2.11}$$

де C > 0, q $\in$ (0,1) – константи.

2.3.2 Числові методи другого порядку. Метод Ньютона та його варіації

Найефективнішим методом розв'язання задач безумовної оптимізації є метод Ньютона, а також його модифікації. Припустимо, що функція f строго опукла і диференційована двічі на $\mathbb{R}^n$, а матриця $f''(x)$ не вироджена на $\mathbb{R}^n$. Послідовність $x^1, x^2, \ldots, x^k, \ldots$ будуватиметься за правилом (2.12)-(2.13):

$$x^{k+1} = x^k + h^k, \tag{2.12}$$

$$h^k = -[f''(x^k)]^{-1} \cdot f'(x^k), \quad k = 0,1, \ldots \tag{2.13}$$

Таким чином зрозуміло, що метод Ньютона є методом мінімізації другого порядку з довжиною кроку $\alpha_k = 1$, а напрямом спадання функції f є вектор h^k. У цьому методі наближення до розв'язку є більшим, ніж у градієнтному, адже квадратична апроксимація заданої функції в малому околі деякої точки значно точніша за лінійну апроксимацію.

Збіжність методу доведено лише для достатньо хорошого початкового наближення x^0 та має проблеми у вигляді великого обсягу обчислень та складності пошуку початкового наближення. Обчислювати та обертати матрицю других похідних цільової функції необхідно на кожному кроці.

Для подолання цих недоліків існують деякі модифікації методи, які прагнуть зберегти переваги у вигляді швидкої збіжності, але при цьому зменшити обсяг обчислень і послабити вимоги до вибору початкового наближення. Є декілька варіантів узагальненого методу, вони відрізняються способом вибору параметра α.

У першому способі вважається, що $\alpha = 1$. Тоді обчислюється точка $x = x^k + \alpha h^k$ і значення функції $f(x) = f(x^k + \alpha h^k)$. Далі перевіряється нерівність (2.14):

$$f(x) - f(x^k) \le \varepsilon\alpha\langle f'(x^k), h^k\rangle, \qquad 0 < \varepsilon < \frac{1}{2} \tag{2.14}$$

де ε – довільна константа, однакова для всіх k.

У другому способі значення α_k обирається як точка мінімуму цільової функції в напрямі антиградієнта. При цьому якщо порівняти два варіанти, очевидно, що перший є більш ефективним, бо у середньому потребує меншої

кількості обчислень цільової функції. Також перевагою методу є те, що для сильно опуклих двічі диференційованих функцій він збігається незалежно від вибору початкової точки x^0 із надлінійною або квадратичною швидкістю.

2.4 Умовна оптимізація

2.4.1 Методи нелінійної умовної оптимізації. Метод проекції градієнта

У рамках цього розділу розглянемо задачу вигляду (2.15)

$$f(x) \to min, x \in X \subset R^n. \tag{2.15}$$

де функція *f(x)* – неперервно диференційована;

множина X – опукла, замкнута та обмежена.

За теоремою Вейєрштрасса (2.16) розв'язок задачі існує:

$$f_* = \inf f(x) > -\infty, \;\; X_* = \{x \in X | f(x) = f_*\} \neq \emptyset. \tag{2.16}$$

Умова опуклості множини X є достатньо загальною. Тому важко очікувати створення ефективних алгоритмів для випадку, коли X – довільна опукла множина. Але існують достатньо прості опуклі множини, такі як паралелепіпед, куля, лінійне різноманіття, для яких задача (2.15) вирішується добре. З іншої сторони, обмеження, що визначають ці прості множини, часто мають реальний економічній або технічний сенс.

Метод проекції градієнта мінімізації функції кількох змінних використовує властивості градієнта, що полягають у тому, що напрямок градієнта співпадає з напрямком найскорішого зростання функції, а напрямок антиградієнта – з напрямком найскорішого спадання.

Ітераційні методи припускають вибір початкового приближення x^0. Однак не існує якихось загальних правил вибору x^0, але, слідуючи з практичного сенсу задачі, доволі часто вдається визначити можливу область розташування точок мінімуму. Тоді початкове наближення необхідно обирати як можна ближче до цієї області.

Розглянемо алгоритм методу проекції градієнта для вирішення задачі (2.15). Проекцією точка *a* на множину $X \in R^n$ називається точка, найближча до точки *a* серед усіх точок з множини *X*.

Число $\pi_x(a)$ є розв'язком задачі проектування (2.17)

$$\varphi(x) = \left||x - a^2|\right| \to \min, \ \ x \in X. \qquad (2.17)$$

Варте уваги те, що проекція точки на множину *X* матиме сенс саме для довільної множини $X \in R^n$, але у загальному випадку проекція точки на множину може визначатися не зовсім однозначно, оскільки задача мінімізації може мати більше одного розв'язку. Існування єдиного такого розв'язку гарантує умова опуклості та замкненості множини $X \in R^n$.

У випадку, якщо *X* – замкнена опукла множина в R^n, то мають місце такі твердження:

а) Точка $\bar{x}$ є проекцією точки *a* на множину *X* ($\bar{x} = \pi x(a)$) тоді і тільки тоді, коли $(\bar{x} - a, x - \bar{x}) \geq 0$ при всіх $x \in X$;

б) для будь-яких точок a^1, $a^2 \in R^n$ виконується оцінка (2.18)

$$||\pi_X(a^1) - \pi_X(a^2)|| \leq \left||a^1 - a^2|\right|. \qquad (2.18)$$

В основу методу проекції градієнта покладено наступну теорему:

Теорема 2.3 Нехай множина $X \in R^n$ – опукла і замкнена, функція *f* – опукла на *X* та диференційована в точці $x^* \in X$. Тоді для того щоб точка x^* була розв'язком задачі (2.4), необхідно і достатньо виконання умови (2.19)

$$x^* = \pi_X\left(x^* - \alpha f'(x^*)\right) \text{ за довільного } \alpha > 0. \quad (2.19)$$

Також у цьому методі у якості чергової точки для наближення до розв'язку обирають проекцію на множину *X* тієї точки, яка отримується за допомогою методу (2.20):

$$x^{k+1} = \pi_X\left(x^k - \alpha_k f'(x^k)\right), \ k = 0,1,2 \ldots \quad (2.20)$$

Важливий момент, що цей метод є збіжним. На кожній k-й ітерації необхідно проектувати точки на множину *X,* а тобто розв'язувати задачу вигляду (2.17) при $a = x^k - \alpha_k f'(x^k)$. У випадках, коли *X* – куля, невід'ємний ортант, паралелепіпед, гіперплощина, півпростір тощо вдається навіть побудувати явну формулу для проекції.

2.5 Висновки

У цьому розділі був виконаний огляд деяких методів оптимізації, їх властивостей, переваг та необхідних умов для правильності та реалістичності функціонування. Вивчені та проаналізовані методи як безумовної, так і умовної оптимізації, їх варіанти застосування та модифікації.

Представлені основні теореми та нерівності, на які можна орієнтуватись при розробці програмного забезпечення та створенні формату вхідних умов. Також було порівняно різні методи для знаходження оптимальних значень розв'язуваної задачі.

3 АНАЛІЗ АРХІТЕКТУРИ СИСТЕМИ РОЗРОБКИ СТРАТЕГІЇ РЕКЛАМНОГО ПРОСУВАННЯ

3.1 Обґрунтування вибору платформи та мови реалізації

Основною платформою для реалізації продукту було обрано операційну систему Windows. Це сучасна платформа з сімейства комерційних операційних систем компанії Microsoft, що орієнтовані на управління за допомогою графічного інтерфейсу. Ця системою є найпопулярнішою з тих, що доступна ринку (у порівнянні з MacOS та Linux), та має менший поріг входу, адже може бути встановлена на будь-якому стандартному комп’ютері.

Для реалізації програмного продукту була обрана мова програмування C# та середовище Visual Studio 2019. C# це потужна об’єктно-орієнтована мова, що швидко розвивається та дуже затребувана в IT-галузі. Серед основних переваг можна назвати такі:

а) Зрозумілий синтаксис;
б) відноситься до мов компілюємого типу;
в) підтримує поліморфізм, наслідування та перевантаження операторів;
г) підтримує статичну типізацію;
д) об’єктно-орієнтований підхід дозволяє вирішувати задачі по побудові масштабних, але в той же час гнучких застосунків.

C# була створена для роботи із фреймворком .NET від компанії Microsoft, що також є перевагою у виборі платформи для реалізації.

3.2 Аналіз вимог користувача до програмного продукту

Для того щоб користувач міг без проблем користуватися системою розробки стратегії вона повинна відповідати таким критеріям.

По-перше, система має базуватися на найпоширеніших показниках рекламного просування, які можна отримати з простих A/B тестів. Правильно аналізувати дані та виводити зрозумілий результат, що базується на часі просування.

По-друге, програма має бути оптимізованою та швидко працювати. Це дає можливість виконувати прорахунки не лише з окремих тестів, а й з довільної множини вибірок.

Останнім критерієм є універсальність. Програма має мати можливість розбивки по каналам просування та категоріям інтересів. Ці категорії можуть складатися як з показників для конкретних авдиторій за інтересами, так і для вікових діапазонів. Таким чином можна прогнозувати просування різноманітної складності.

Таким чином можна узагальнити вимоги, описані вище, та зробити висновок про те, що застосунок має бути гнучким до сприймання вхідних даних та швидко й точно виконувати аналіз показників.

3.3 Аналіз архітектури системи розробки рекламної стратегії

Програмний продукт містить у собі такі модулі:

а) TextDataClass – абстрактний клас, що відповідає за читання/запис текстових даних;

б) Channel – клас, що відповідає за канали просування;

в) CustomerCategory – клас, що відповідає за категорію клієнтів;

г) LinearSystem = вирішення матричних рівнянь;

д) Program – головна точка входу;

е) Optimization – метод оптимізації

ж) AuxiliaryFuction – значення за замовчуванням;

з) Decomposition – розклади квадратної матриці;

и) SimpleAction – прості дії, такі як перетворення і т.д.

к) Projection – пошук проекції на гіперплощину;

л) Vector – робота із векторами;

м) Matrix – робота із матрицями;

н) BestPromotion – цільовий клас стратегії;

о) Form1 – графічний інтерфейс.

Діаграма класів та модулів, що відображає вміст та структуру програми відображена на рисунку 3.1:

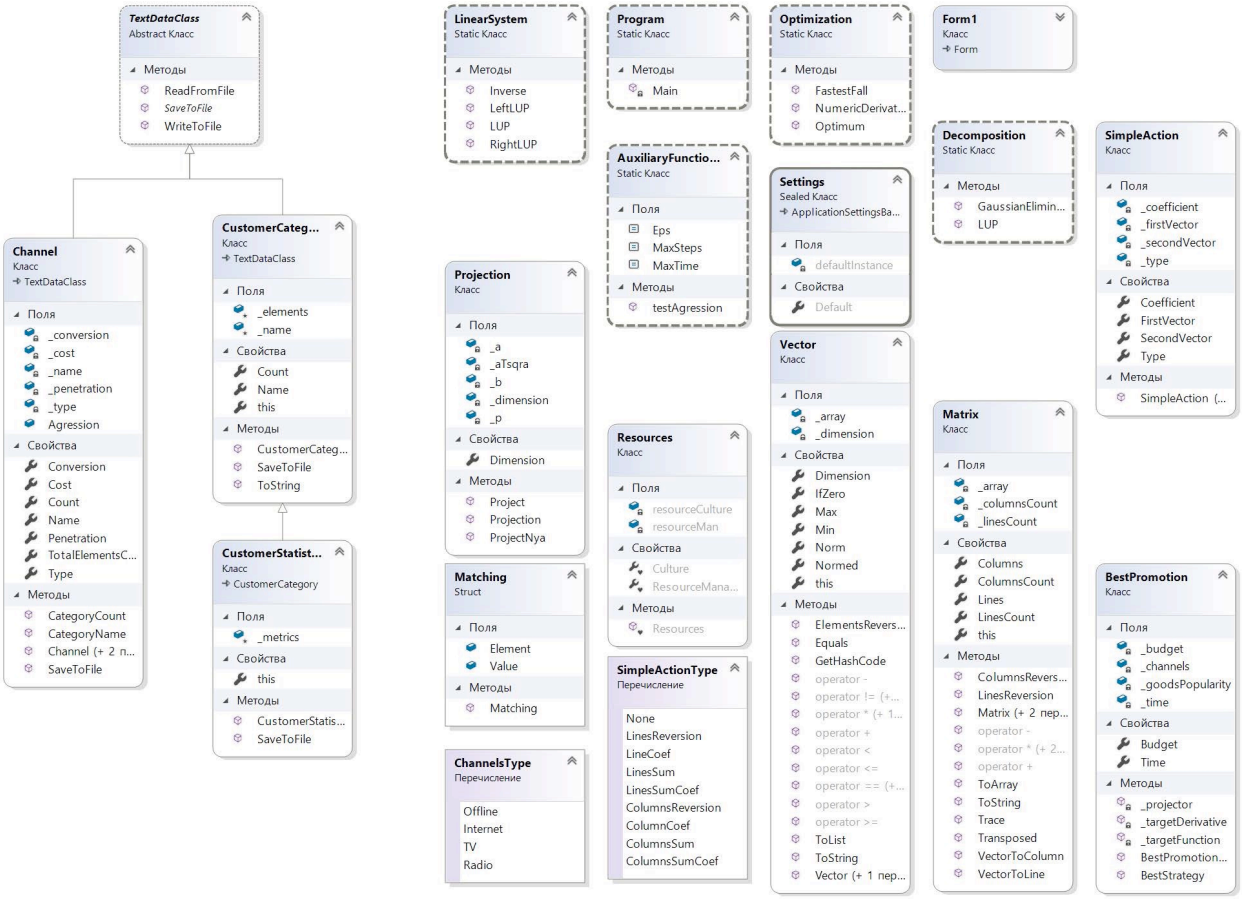


Рисунок 3.1 - Діаграма класів та модулів

3.4 Алгоритм роботи системи розробки стратегії

Першими кроками роботи програми є ініціалізація та завантаження даних про канали, категорії та рекламних показників за авдиторіями.

Для обраної моделі головною фігурою виступає багатогранник — сімплекс. Для розрахунків вводимо функцію, яка є опуклою на певних відрізках. У цьому випадку використовуємо умову Ліпшиця, яка гарантує сходження методу із швидкістю геометричної прогресії. Після цього задаємо грані опорними векторами $\bar{x}$, $\bar{y}$ та $\bar{z}$.

Таким чином ми задаємо гіперплощину. Кожна змінна часу для кожного каналу задовольняє умовам (3.3). Щоб забезпечити гарантоване знаходження результату необхідно використати метод найскорішого спуску з використанням методу золотого перетину. На кожному кроці будемо робити проекцію на гіперплощину та вимірювати відстань від точки, створеної на минулій ітерації. Після вирішення задачі оптимізації ми отримуємо максимальне допустиме значення часу, що було витрачено на просування за кожним з напрямків. Таким чином ми отримуємо результат найоптимальнішого просування, що базується на часі просування.

Вхідні дані щодо каналів, категорій та інших показників ефективності зчитуються з файлів розширення .txt, шлях до яких обирається безпосередньо у вікні програми.

Усі вхідні значення розглядаються як три виміри: канали - перший вимір, категорії - другий, елементи - третій. Ітераторами вимірів виступають i, j, та k. Для розрахунку доходу використовуємо суму по i (за всіма каналами), та суму по j,k (за всіма елементами всіх категорій) (3.1).

$$\sum R_{ijk} Cv_{ijk} Pop_{jk} T_{ijk}\, A_i\left(T_{ijk}\right). \tag{3.1}$$

де R_{ijk} - охоплення за час у розрізі по елементам категорії для кожного каналу.

Cv_{ijk}- конверсія за час у розрізі (умовні гроші/одиницю охвату)

Pop_{jk} - популярність товару серед елементів різних категорій

T_{ijk} - час просування у розрізі

A_i- функція агресії

За обмеження будемо використовувати (3.2) і (3.3):

$$c_i P_{ijk} T_{ijk} < C, \tag{3.2}$$

$$T_0 > T_k > T. \tag{3.3}$$

де c_i – вартість;

C – бюджет.

Значення T_{ijk}знаходяться у вирішенні задачі оптимізації. При знаходженні кожного елементу ми можемо переходити до циклу та продовжувати вирішувати задачу без його наявності. За умови зупинки процесу оптимізації можна використовувати (3.4) і (3.5):

$$\left||x_i - x_{i+1}|\right| < \varepsilon, \tag{3.4}$$

$$|f(x_i) - f(x_{i+1})| < \varepsilon. \tag{3.5}$$

де ε – константа, що приймається за 0,0001.

Так як у ході вирішення задачі оптимізації можуть виникати випадки переваги та розтягування по одній з осей, доцільно використовувати обидві вимови для знаходження моменту зупинки алгоритму.

Послідовність кроків виконання програми, зображена у функціональній схемі на рисунку 3.2:

Рисунок 3.2 - Функціональна схема програми, аркуш 51

3.5 Керівництво користувача

Для користування програмою користувача необхідно запустити виконуваний файл .exe, у якому містить програмний продукт. Далі відобразиться графічний інтерфейс програми (рисунок 3.3), у якому маються поля для обирання шляху до вхідних даних (мають міститися у текстовому файлі формату .txt).

Рисунок 3.3 - Графічний інтерфейс програмного продукту, аркуш 52

Далі необхідно послідовно вказати шлях до файлу із категоріями та каналами просування. При цьому порядок завантаження не має значення. Після вказання необхідно натиснути на кнопки «Додати канал» та «Додати категорію» відповідно. Користувач може завантажувати необмежену кількість каналів та категорій у програмний продукт. Після додавання вони відображаються у правій секції програмного продукту.

Вигляд файлу із вхідними даними має такий вигляд як на рисунку 3.4 та рисунку 3.5 відповідно.

```
Facebook
Internet
120
<new Category>
Стать
чоловіча
0,4
0,07
жіноча
0,6
0,07
<new Category>
Вік
до18
0,2
0,05
від18до30
0,3
0,09
від30до55
0,4
0,07
від55
0,1
0,05
```

Рисунок 3.4 - Приклад вхідного файлу із каналом просування, аркуш 53

```
Вік
до18
0,3
від18до30
0,6
від30до55
0,05
від55
0,05
```

Рисунок 3.5 - Приклад вхідного файлу із категорією просування, аркуш 53

Після натискання кнопки «Старт» програма виконує розрахунки та заповнює колонку із розрахунковим часом просування (рисунок 3.6), де останній вимірюється в умовних одиницях, що дорівнюють одному року.

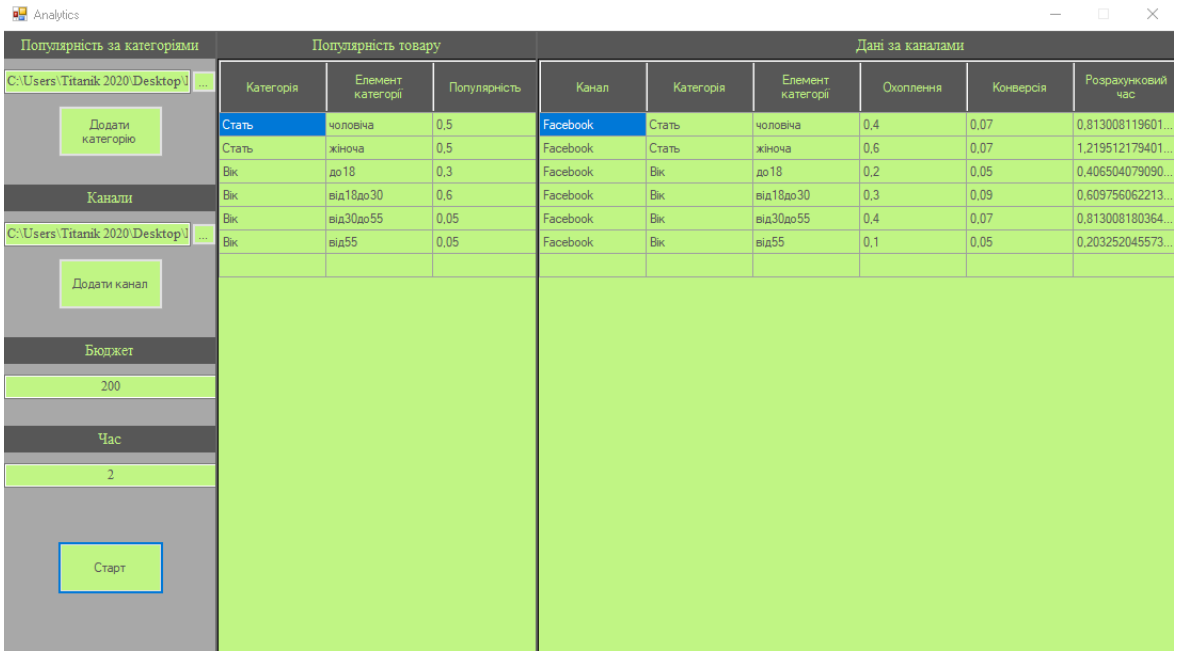


Рисунок 3.6 - Приклад роботи програми, аркуш 54

Для закінчення роботи з програмою її необхідно закрити самостійно.

3.6 Аналіз результатів, отриманих в роботі

3.6.1 Опис та аналіз вибірки вхідних даних

Для ефективної та правильної роботи програмного продукту вхідні дані задаються у певному порядку. Для зручності використовується підготовлений файл, у якому змінюються лише конкретні значення рекламних показників. Приклад вхідних даних за каналом представлений у файлі Channel_FB_example.txt, а приклади за категоріями у файлах Cat_age_example.txt та Cat_Sex_example.txt

У файлі з категоріями відмічається назва, витрачений бюджет та показники за статевими та віковими ознаками. У файлі з каналами вони мають такий порядок:

- Показник охоплення;
- Показник конверсії, визначається за формулою (3.1).

$$Cv = \frac{Q_к}{Q_з} \cdot 100\%. \quad (3.5)$$

де $Q_к$ – кількість користувачів, які виконали цільову дію;

$Q_з$ – загальна кількість всіх користувачів.

У файлі з категоріями використовується лише показник популярності, що визначається за формулою (3.6)

$$P = \frac{Q_r}{R}. \quad (3.6)$$

де Q_r – загальна кількість всіх реакцій;

R – охоплення.

3.6.2 Дослідження практичних результатів

Для перевірки якості результатів у ідеальних умовах необхідно створити рекламну кампанію, провести тести і виконати масштабування. Ці дії передбачають залучення бюджету та не є доступними. Тому було вирішено перевірити результати на існуючих прикладах рекламного агенства, де за вхідні дані та результат прорахунку було взято приклад на рисунку 3.7.

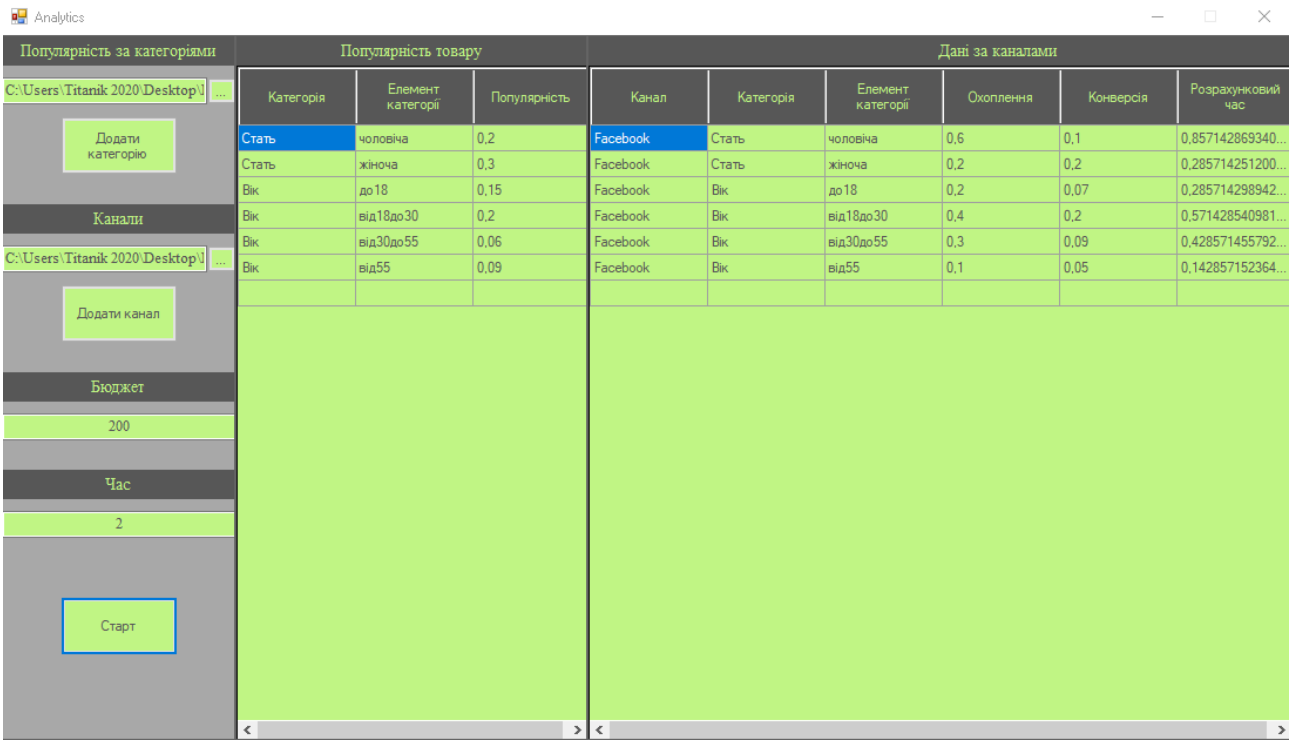


Рисунок 3.7 - Вхідні дані для дослідження ефективності, аркуш 56

Далі виконувався поетапний аналіз за групами оголошень, що просувалася по авдиторіям з віковим діапазаном «до 18» і «від 30 до 55». Як критерій оцінки якості використувався показник популярності та його зміни. Результати дослідження наведено у таблиці 3.1 та таблиці 3.2.

Таблиця 3.1 – Дослідження показників рекламного просування «до 18»

Час просування	Показник популярності	Показник конверсії
1 місяць	0,15	0,07
2 місяці	0,14	0,07
3 місяці	0,14	0,05
4 місяці	0,09	0,03
5 місяців	0,05	0,03
6 місяців	0,03	0,02

Таблиця 3.2 – Дослідження показників рекламного просування «від 30 до 55»

Час просування	Показник популярності	Показник конверсії
1 місяць	0,06	0,09
2 місяці	0,07	0,07
3 місяці	0,06	0,07
4 місяці	0,05	0,05
5 місяців	0,05	0,02
6 місяців	0,03	0,01

Проаналізувавши результати, можна помітити, що показники популярності конверсії починають змінюватися до низу через 5 місяців у першому випадку та через 3 місяці у другому. Прорахунок за цими рекламними авдиторіями визначав просування терміном у 4 та 3 місяці відповідно. Таким чином визначено, що прораховані показники є реалістничними та оптимальними.

3.7 Висновки

У третьому розділі було детально розглянуто архітектуру та склад програмного продукту. Для реалізації було обране комерційне сімейство операційних систем Windows. У якості мови програмування – об’єктно-орієнтована мова C#. Був проведений детальний опис їх переваг та можливостей.

Були визначені усі модулі, які мають бути розроблені для повної реалізації поставленої задачі. Розроблене детально керівництво користувача для комфортної роботи з програмним продуктом.

Також було проаналізовано формат вхідних даних та описано їх складові. У заключенні відображений аналіз прорахованих показників.

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

4.1 Постановка задачі проектування

Спроектувати програмний продукт для виведення опорних даних, що використовуються при розробці стратегії рекламного просування у сфері інтернет-маркетингу. Продукт призначений для використання на операційній системі Windows та потребує наявності початкових даних, що отримуються методом спліт-тестування рекламних оголошень.

4.2 Обґрунтування функцій та параметрів програмного продукту

Виходячи з конкретних цілей, які реалізуються :

F1 – завантаження вхідних даних: а) введення у код програми, б) завантаження з пам’яті пристрою.

F2 – категоризація даних - а) повне розподілення у програмі, б) розподілення з одного окремого файлу, в)розподілення з двох окремих файлів, г) комбінація по всім варіантам.

F3 – оптимізація параметрів–а) умовна оптимізація довільної функції, б) умовна оптимізація опуклої функції.

F4 – отримання результатів оптимізації - а) відображення результатів за часом просування, б) відображення результатів за витраченим бюджетом.

F5– відображення результатів роботи - а) створення слайду презентації б) лише вивід користувачу.

Виходячи з представлених варіантів будуємо морфологічну карту (рисунок 4.1).

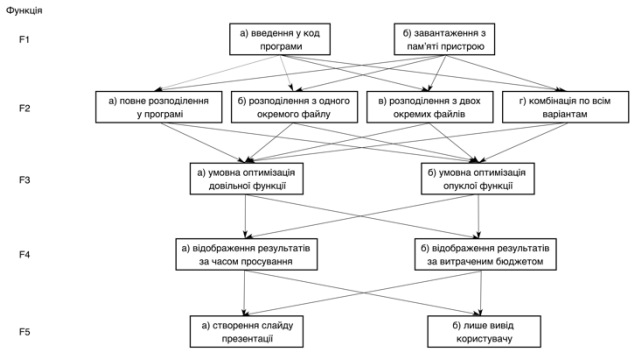


Рисунок 4.1 - Морфологічна карта, аркуш 60

Спираючись на карту була побудована позитивно-негативна матриця (таблиця 4.1)

Таблиця 4.1 Позитивно-негативна матриця

Основна функція	**Варіант реалізації**	**Переваги**	**Недоліки**
F1	А	Відсутність формату вводу	Вищий поріг входу користувача
	Б	Зручність, структуризація даних, можливість збереження	Необхідність вивчення формату вводу даних
F2	А	Швидкість зміни розподілення	Відсутність запам’ятовування та необхідність повторного вводу
	Б	Зручність	Відсутність гнучкості
	В	Гнучкість реалізації	Необхідність парного зберігання файлів
	Г	Швидкість підготовки даних	Велика кількість зайвих наборів даних
F3	А	Легкість оптимізації	Нереалістичність умов моделювання
	Б	Точність результатів	Вища складність проектування
F4	А	Оптимальність параметру при розробці стратегії	Недоступність розбивки по бюджету

	Б	Наочність витрат бюджету	Фантомність параметру у рамках розробки стратегії
F5	А	Зручнеоформлення результатів роботи	Не використовується у початковому вигляді
	Б	Зручність аналізу	Необхідно інтерпретувати у потрібний вигляд самостійно

Для характеристики прототипу програмного додатку використовуємо параметри X1 – X5. На основі даних, що представлені у літературі, визначаємо мінімальні, середні отримуванні та максимально допустимі значення (таблиця 4.2)

Таблиця 4.2 Система параметрів додатку

Найменування параметру	Позначення параметру	Значення параметру		
		Мінімальне	Середнє	Максимальне
Час розробки, людина*год	X1	340	420	500
Час роботи алгоритму, мс	X2	100	250	400
Рекомендована частота процесору, ГГц	X3	1,7	2	2,3
Час обробки результату, мс	X4	8	14	20
Рекомендована швидкість запису на диск, МБ/с	X5	0	32	64

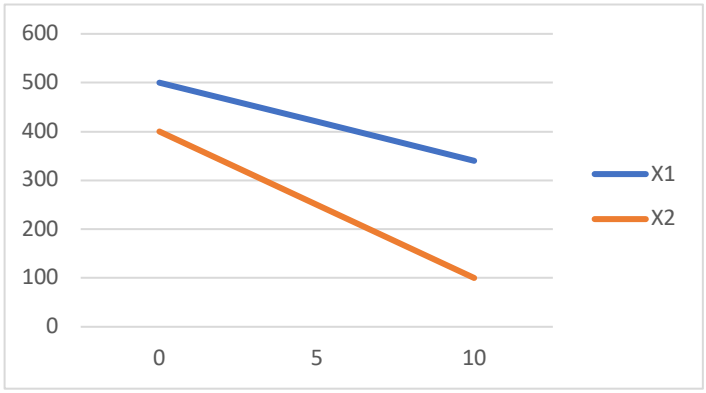


Рисунок 4.2 - Значення параметрів X1, X2

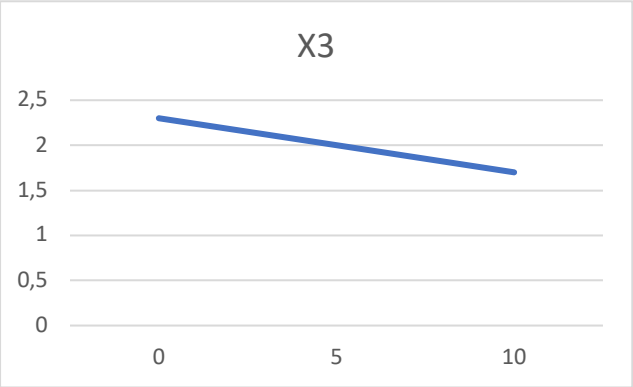


Рисунок 4.3 - Значення параметру X3

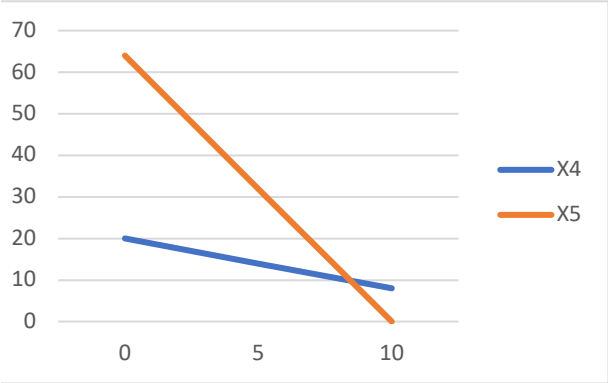


Рисунок 4.4 - Значення параметрів X4, X5

Вагомість параметрів оцінюється за допомогою методів попарного зрівняння. Ранги варіюються від 1 до 5 (вищий - нижчий). Результати наведені в таблицях 4.3-4.4

Таблиця 4.3 Результат оцінки параметрів

Параметр	Ранг параметру по оцінці експерта							Сума рангів, R_i	ВідхиленняΔ_i	Квадрат відхилення, $(\Delta_i)^2$
	1	2	3	4	5	6	7			
X1	2	1	2	2	1	2	3	13	-8	64
X2	3	3	3	1	3	3	1	17	-4	16
X3	1	2	1	3	2	1	2	12	-9	81
X4	4	4	4	4	4	4	5	29	8	64
X5	5	5	5	5	5	5	4	34	13	169
Разом	15	15	15	15	15	15	15	105	0	394

Табл. 4.4 Попарне зрівняння параметрів

Параметри	**Експерти**							**Кінцева оцінка**	**Числове значення**
	1	**2**	**3**	**4**	**5**	**6**	**7**		
X1 та X2	>	>	>	<	>	>	<	>	1.5
X1 та X3	<	>	<	>	>	<	<	<	0.5
X1 та X4	>	>	>	>	>	>	>	>	1.5
X1 та X5	>	>	>	>	>	>	>	>	1.5

X2 та X3	<	<	<	>	<	<	>	<	0.5
X2 та X4	>	>	>	>	>	>	>	>	1.5
X2 та X5	>	>	>	>	>	>	>	>	1.5
X3 та X4	>	>	>	>	>	>	>	>	1.5
X3 та X5	>	>	>	>	<	>	>	>	1.5
X4 та X5	>	>	>	>	>	>	<	>	1.5

Визначимо коефіцієнт конкординації: $W = \frac{12S}{N^2(n^3-n)} = \frac{12*394}{7^2(5^3-5)} = 0.80 > W_k = 0.67$

Так як коефіцієнт конкординації більше нормативного, результати вважають достовірними.

Розрахунок вагомості параметрів наведено в таблиці 4.5

Таблиця 4.5 Розрахунок вагомості параметрів

Параметри	Параметри xj					Перший крок		Другий крок		Третій крок	
	X1	X2	X3	X4	X5	b_i	$K_{вi}$	b_i	$K_{вi}$	b_i	$K_{вi}$
X1	1	1,5	0,5	1,5	1,5	6	0,23	29	0,24	131,5	0,24
X2	0,5	1	0,5	1,5	1,5	5	0,19	23,5	0,20	105,25	0,19
X3	1,5	1,5	1	1,5	1,5	7	0,27	35,5	0,29	163,75	0,30
X4	0,5	0,5	0,5	1	1,5	5	0,19	18,5	0,15	84,25	0,15
X5	0,5	0,5	0,5	0,5	1	3	0,12	14,5	0,12	67,75	0,12
Загалом:						26	1	121	1	552,5	1,00

Враховуючи дані з порівнянь варіантів реалізацій функцій можна виключити з реалізацій функцій наступні варіанти: F2(а, б, г), F3(а), F5(а)
Залишаються наступні варіанти:

1. F1(а)=>F2(в)=>F3(б)=>F4(а)=>F5(б)
2. F1(б)=>F2(в)=>F3(б)=>F4(а)=>F5(б)
3. F1(а)=>F2(в)=>F3(б)=>F4(б)=>F5(б)
4. F1(б)=>F2(в)=>F3(б)=>F4(б)=>F5(б)

Таблиця 4.6 Доступні варіанти реалізації

Основна функція	Варіант реалізації	Абсолютне значення параметру	Бальна оцінка параметру	Коефіцієнт вагомості параметру	Коефіцієнт якості
F1	а)	440	3,75	0,24	0,90
	б)	430	4,37	0,24	1,05
F2	в)	150	8,33	0,19	1,58
F3	б)	1,9	6,66	0,30	1,20
F4	а)	10	8,33	0,15	1,25
	б)	18	1,66	0,15	0,25
F5	б)	1	9,84	0,12	1,18

Обрахуємо коефіцієнти якості кожного з варіантів розробки: -

$$K_{я1} = 0{,}90 + 1{,}58 + 1{,}20 + 1{,}25 + 1{,}18 = 6{,}11$$

$$K_{я2} = 1{,}05 + 1{,}58 + 1{,}20 + 1{,}25 + 1{,}18 = 6{,}26$$

$$K_{я3} = 0{,}90 + 1{,}58 + 1{,}20 + 0{,}25 + 1{,}18 = 5{,}11$$

$$K_{я4} = 1{,}05 + 1{,}58 + 1{,}20 + 0{,}25 + 1{,}18 = 5{,}26$$

Оскільки варіант 2 має найбільший коефіцієнт якості, він є найкращим.

4.3 Економічний аналіз варіантів розробки

Для оцінки трудомісткості розробки спочатку проведемо розрахунок трудомісткості. Усі варіанти мають наступні основні завдання:

1) Розподілення за каналами і категоріями з двох окремих файлів
2) Умовна оптимізація опуклої функції
3) Вивід результатів на екран

Також кожний з варіантів має два додаткових завдання, які є реалізаціями розгалужених варіантів розробки незалежного модуля. Далі

наведено варіанти додаткових завдань(два завдання, які мають номери 4 в реалізаціях та два завдання, які мають номери 5 в реалізаціях)

4.1) Введення у код програми

4.2) Завантаження з пам’яті пристрою

5.1) Відображення результатів за часом просування

5.2) Відображення результатів за витраченим бюджетом

В варіанті 1 присутні наступні додаткові завдання під номерами 4.1 та 5.1

В варіанті 2 присутні наступні додаткові завдання під номерами 4.2 та 5.1

В варіанті 3 присутні наступні додаткові завдання під номерами 4.1 та 5.2

В варіанті 4 присутні наступні додаткові завдання під номерами 4.2 та 5.2

За ступенем новизни до групи Б відноситься завдання 4.1, до групи В відносяться завдання 1, 3, до групи Г завдання 2, 5.2, 5.1, 4.2

За складністю алгоритмів до групи 1 відноситься завдання 1, до групи 2 відноситься завдання 2, 5.2, 4.1, 5.1 до групи 3 відноситься завдання 3, 4.2

Спираючись на норми розрахункового часу визначимо трудомісткість. Вона складає для першого завдання $T_р$=43 людино-днів. Поправочний коефіцієнт складає $К_п$=0,81 (нормативно-довідкова інформація). Оскільки під час виконання даного завдання використовуються новостворені модулі, врахуємо це за допомогою коефіцієнта $К_{ст}$ = 0,6. Коефіцієнти $К_м$ і $К_{ст.п}$, які враховують відповідно програмування на мові низького рівня та розробку стандартного програмного забезпечення, для всіх завдань дорівнюють 1.

Повна трудомісткість першого завдання(складність – 1, новизна – В):

$$T_1 = 43 * 0{,}81 * 0{,}6 = 20{,}90$$

Аналогічно для завдання 2, 5.2(складність – 2, новизна – Г): –

$T_р$= 12; $К_п$ = 0,43; $К_{ст}$ = 0,8; $T_2 = 12 * 0{,}43 * 0{,}8 = 4{,}13$

Аналогічно для завдання 3(складність – 3, новизна – В):

$T_р$= 12; $К_п$ = 0,6; $К_{ст}$ = 0,6; $T_3 = 12 * 0{,}6 * 0{,}6 = 4{,}32$

Аналогічно для завдання 4.1(складність – 2, новизна – Б)

$T_р$= 27; $К_п$ = 1,08; $К_{ст}$ = 0,6; $T_4 = 27 * 1{,}08 * 0{,}6 = 17{,}50$

Аналогічно для завдання 5.1(складність – 2, новизна – Г):

$T_р$= 12; $К_п$ = 0,43; $К_{ст}$ = 0,8; $T_5 = 12 * 0{,}43 * 0{,}8 = 4{,}13$

Аналогічно для завдання 4.2(складність – 3, новизна – Г)

$T_р$= 8; $К_п$ = 0,36; $К_{ст}$ = 0,8; $T_6 = 8 * 0{,}36 * 0{,}8 = 2{,}30$

Визначимо повну трудомісткість варіантів(людино-днів):

$$T_1 = 20{,}90 + 4{,}13 + 4{,}32 + 17{,}50 + 4{,}13 = 50{,}98$$

$$T_2 = 20{,}90 + 4{,}13 + 4{,}32 + 2{,}30 + 4{,}13 = 35{,}78$$

$$T_3 = 20{,}90 + 4{,}13 + 4{,}32 + 17{,}50 + 4{,}13 = 50{,}98$$

$$T_4 = 20{,}90 + 4{,}13 + 4{,}32 + 2{,}30 + 4{,}13 = 35{,}78$$

Найбільш трудомістким завданням є 5.1, найбільш трудомісткий варіант - 2

Далі вважається, що робочий день складає 8 годин, в тижні п’ять робочих днів. В розробці бере участь один програміст з окладом 16000 грн та тестувальник з окладом 10000 грн. Визначимо середню заробітну плату за годину:

$$С_ч = \frac{16000 + 10000}{2 * 22 * 8} = 73{,}86$$

Тоді заробітна плата для кожного з варіантів реалізації(грн):

1) $С_{ЗП} = 73{,}86 * 8 * 50{,}98 = 30123{,}06$
2) $С_{ЗП} = 73{,}86 * 8 * 35{,}78 = 21141{,}69$
3) $С_{ЗП} = 73{,}86 * 8 * 50{,}98 = 30123{,}06$
4) $С_{ЗП} = 73{,}86 * 8 * 35{,}78 = 21141{,}69$

Відрахування на соціальне страхування (грн):

1) $С_{ВІД} = 30123{,}06 * 0{,}22 = 6627{,}07$
2) $С_{ВІД} = 21141{,}69 * 0{,}22 = 4651{,}17$
3) $С_{ВІД} = 30123{,}06 * 0{,}22 = 6627{,}07$
4) $С_{ВІД} = 21141{,}69 * 0{,}22 = 4651{,}17$

Далі розрахуємо витрати на оплату однієї машино-години. Враховуючи, що вона обслуговує одного спеціаліста з окладом 16000 грн та одного з окладом 10000 грн, то для двох машин отримаємо

$$С_г = 12 * 16000 * 0,6 + 12 * 10000 * 0,6 = 187200 \text{ грн}$$

Враховуючи додаткову заробітну плату40%

$$С_{ЗП} = 187200 * (1 + 0,4) = 262080$$

Відрахування на соціальне страхування22%

$$С_{ВІД} = 262080 * 0,22 = 57657,6$$

Розрахуємо амортизаційні підрахунки (амортизація 25%, вартість ЕОМ 28000 грн)

$$С_А = К_{ТМ} * К_А * Ц_{ПР} = 1,15 * 0,25 * 28000 = 8050 \text{ грн}$$

Розрахуємо витрати на ремонт та профілактику:

$$С_Р = К_{ТМ} * Ц_{ПР} * К_Р = 1,15 * 28000 * 0,05 = 1610 \text{ грн}$$

Розрахуємо ефективний годинний фонд часу ПК за рік

$$Т_{ЕФ} = (365 - 142 - 16) * 8 * 0,8 = 1324,8 \text{ год}$$

Розрахуємо витрати на електроенергію

$$С_{ЕЛ} = 1324,8 * 0,6 * 0,8 * 1,75 = 1112,83 \text{ грн}$$

Накладні витрати рівні:

$$С_Н = 28000 * 0,67 = 18760 \text{ грн.}$$

Отже експлуатаційні витрати(грн):

$$С_{ЕКС} = 262080 + 57657,6 + 8050 + 1610 + 1112,83 + 18760 = 349270,43$$

Тоді собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М-Г} = \frac{349270,43}{1324,8} = 263,64 \text{ грн/год}$$

Враховуючи, що всі роботи ведуться на ЕОМ, витрати на оплату машинного часу:

1) $С_м = 263,64 * 8 * 50,98 = 107522,94$
2) $С_м = 263,64 * 8 * 35,78 = 75464,31$
3) $С_м = 263,64 * 8 * 50,98 = 107522,94$
4) $С_м = 263,64 * 8 * 35,78 = 75464,31$

Накладні витрати відповідно

1) $С_Н = 107522,94 * 0,67 = 72040,34$
2) $С_Н = 75464,31 * 0,67 = 50561,09$
3) $С_Н = 107522,94 * 0,67 = 72040,34$
4) $С_Н = 75464,31 * 0,67 = 50561,09$

Розрахуємо повну вартість розробки за варіантами:

1) $С_{ПП} = 30123,06 + 6627,07 + 107522,94 + 72040,34 = 216313,41$
2) $С_{ПП} = 21141,69 + 4651,17 + 75464,31 + 50561,09 = 151818,26$
3) $С_{ПП} = 30123,06 + 6627,07 + 107522,94 + 72040,34 = 216313,41$
4) $С_{ПП} = 21141,69 + 4651,17 + 75464,31 + 50561,09 = 151818,26$

4.4 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня

$$К_{ТЕР1} = \frac{6,11}{216313,41} = 2,82 * 10^{-5}$$

$$К_{ТЕР2} = \frac{6,26}{151818,26} = 4,12 * 10^{-5}$$

$$К_{ТЕР3} = \frac{5,11}{216313,41} = 2,36 * 10^{-5}$$

$$К_{ТЕР4} = \frac{5,26}{151818,26} = 3,46 * 10^{-5}$$

4.5 Висновки

Отже враховуючи всі дослідження, що описані вище, можна сказати, що 2 варіант реалізації є найбільш оптимальним зі сторони якісно-економічної оцінки.

Його коефіцієнт техніко-економічного рівня складає $4{,}12*10^{-5}$.

Розробка цього варіанту передбачає такі обов’язкові завдання як:

а) Розподілення за каналами і категоріями з двох окремих файлів;

б) умовна оптимізація опуклої функції;

в) вивід результатів на екран.

Серед завдань між якими ставився вибір в даному варіанті реалізовані такі завдання:

а) Завантаження з пам’яті пристрою;

б) відображення результатів за часом просування.

ВИСНОВКИ

Основною задачею дипломної роботи була створення системи розробки стратегії рекламного просування у сфері інтернет-маркетингу. В результаті роботи були отримані наступні результати:

а) проведено дослідження сучасної ситуації у сфері інтернет-маркетингу, виконано аналіз доступних платформ та їх інструментів, що надають можливість аналізувати та інтерпретувати рекламні показники;

б) детально розглянуто та описано алгоритми безумовної та умовної оптимізації, визначено їх переваги та недоліки, наведено основні теореми та основні критерії розв’язку задачі, а саме: збіжність методів, слідування умові Ліпшица, опуклість функцій, проекція градієнта, метод найскорішого спуску. Досліджено процес оптимізації рекламних показників та методи їх інтерпретації;

в) у практичній частині роботи був обраний метод умовної оптимізації за допомогою проекції градієнта та використання методу найскорішого спуску з використанням методу золотого перетину. Також били визначені основні умови для розв’язку задачі та описані задані обмеження;

г) на базі створеної архітектури була розроблена допоміжна система розробки рекламної стратегії на базі комерційного сімейства операційних систем Windows.

д) недоліками системи є необхідність у вивчені конкретних ситуацій та авдиторій, а також розробка для них функцій агресій. Проте, у перспективі можливо задіяти нейромережі для досягнення вищого рівня автоматизаціх та прямої взаємодії із інструментами рекламного кабінету Facebook.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бартіш М.Я., Дудзяний І.М. Дослідження операцій. Частина 4 : Нелінійне програмування : підручник. Львів: ЛНУ ім. Івана Франка, 2011. 208 с.

2. Яковлева А.П., Спекторський І.Я. Методичний посібник до лабораторних робіт з курсу «Методи оптимізації». Для студентів спеціальності: системний аналіз і управління: навч.-метод. Посіб. Київ: НТУУ «КПІ», ННК «ІПСА», 2004. 64 с.

3. Сергеев Я.Д., Квасов Д.Е. Краткое введение в теорию липшицевой глобальной оптимизации: учеб.-метод. пособие. Нижний Новгород: Изд-во ННГУ, 2016. 48 с.

4. The ultimate guide to Facebook marketing. URL: https://www.canva.com/learn/facebook-marketing-guide/ (дата звернення: 25.05.2020)

5. Введение в C#. Язык C# и платформа .NET Core. URL: https://metanit.com/sharp/tutorial/1.1.php (дата звернення: 25.05.2020)

6. Метод проекции градиента решения задачи многомерной условной минимизации. URL: http://mf.grsu.by/Kafedry/prikl_mat/academic_process/045/044 (дата звернення: 26.05.2020)

7. SMM в 2019 году. URL: https://vc.ru/marketing/46331-smm-v-2019-godu (дата звернення: 26.05.2020)

8. История интернет-рекламы. URL: https://andreyex.ru/prodvizhenie-sajta/istoriya-internet-reklamy/ (дата звернення: 26.05.2020)

9. История рекламы – от древности и до настоящего момента. URL: https://znaytovar.ru/s/Istoriya-reklamy--ot-drevnosti.html (дата звернення: 27.05.2020)

10. Типы рекламных кампаний в Facebook и Instagram. URL: https://netpeak.net/ru/blog/tipy-reklamnykh-kampaniy-v-facebook-i-instagram/ (дата звернення: 27.05.2020)

11. Алгоритми Facebook. Як з ними працювати SMM-спеціалістам. URL: https://creativesmm.com.ua/algoritmi-facebook-jak-z-nimi-pracjuvatu-smm-specialistam/ (дата звернення: 27.05.2020)

12. Nielsen Holdings. URL: https://uk.wikipedia.org/wiki/Nielsen_Holdings (дата звернення: 27.05.2020)

13. Ліпшицеве відображення. URL: https://uk.wikipedia.org/wiki/Ліпшицеве_відображення (дата звернення: 28.05.2020)

14. Основны терміни SMM-маркетолога. Основні поняття. Частина 1. URL: https://www.imena.ua/blog/terms-for-smm-part-1/ (дата звернення: 28.05.2020)

15. Метод градиентного спуска. URL: http://www.machinelearning.ru/wiki/index.php?title=Метод_градиентного_спуска (дата звернення: 28.05.2020)

16. Метод наискорейшего спуска. URL: https://poznayka.org/s7159t1.html (дата звернення: 28.05.2020)

17. Метод золотого сечения. Симметричные методы. URL: http://www.machinelearning.ru/wiki/index.php?title=Метод_золотого_сечения._Симметричные_методы (дата звернення: 28.05.2020)

18. Метод золотого сечения. URL: https://cpp.mazurok.com/tag/метод-золотого-сечения/ (дата звернення: 28.05.2020)

ДОДАТОК А

Система розробки стратегії просування у сфері інтернет-маркетингу

Терещенко Антон В'ячеславович
КЕРІВНИК: Древаль Максим Михайлович

Мета та цілі

- Розглянути доступні методи для створення системи розробки рекламної стратегії
- Провести огляд існуючих інструментів та показників рекламного кабінету Facebook
- Розробити програмний продукт, що буде базуватися на актуальних показниках та визначатиме опорні показники для комерційної пропозиції

Актуальність

- 1,6 млрд. переглядів кожного дня
- Гнучка система комунікації зі споживачами
- Архітектура імідж- та бізнес-складових
- Платформа швидкого доступу з низьким порогом входу

Підхід до оптимізації

- **Градієнтний метод**

Послідовність наближень змінної x до точки мінімуму:

$$x^{k+1} = x^k - \alpha_k f'(x^k), \qquad \alpha_k > 0, \qquad k = 0,1,2,\dots$$

- **Метод проекції градієнта**
- **Метод найскорішого спуску із використанням методу золотого перетину**

Засоби для розробки

Операційна система	Windows
Мова програмування	C#
Середовище розробки	Microsoft Visual Studio 2019

- **Зрозумілий синтаксис**
- **Мова компілюємого типу**
- **Підтримує поліморфізм, наслідування та перевантаження операторів**
- **Підтримує статичну типізацію**
- **Базується на об'єктно-орієнтованому підході**

Етапи роботи програми

Показники вхідної вибірки

- **Охоплення** R
- **Конверсія** $Cv = \frac{Q_к}{Q_з} \cdot 100\%$
- **Популярність** $P = \frac{Q_r}{R}.$

$Q_к$ – кількість користувачів, які виконали цільову дію;
$Q_з$ – загальна кількість всіх користувачів.
Q_r – загальна кількість всіх реакцій;
R – охоплення

Вхідні дані за каналами

```
Facebook
Internet
120
<new Category>
Стать
чоловіча
4
7
жіноча
6
7
<new Category>
Вік
до18
2
5
від18до30
4
9
від30до55
4
7
від55
1
5
```

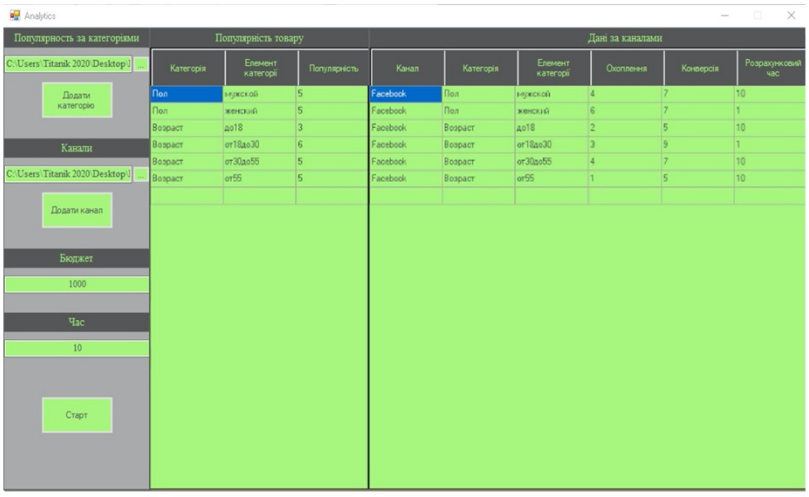
- Назва каналу
- Платформа просування
- Бюджет на канал
- Категорія розмежування авдиторії
- Показник охоплення
- Показник конверсії

Вхідні дані за категоріями

```
Вік
до18
3
від18до30
6
від30до55
5
від55
5
```

- Тип категорії
- Назва категорії
- Показник популярності товару

Приклад роботи програми



Результати роботи

- Програма розпізнає базові рекламні показники рекламного кабінету
- Можливе створення вибірок за різними комбінаціями каналів та категорій просування. Для цього реалізована можливість завантаження з різних файлів
- У результаті отримується значення оптимального просування за часом, де найбільш час просування по авдиторії є найбільш оптимальним варіантом

Фактичний аналіз прогнозу

- **Показники A/B тестування:**

  Популярність: 0,15

  Конверсія: 0,07

Прогноз часу просування: 5 місяців

Авдиторія: до 18

- **Показники завершеного просування за прогнозом:**

Час просування	Популярність (P)	Конверсія (Cv)	Δ P	Δ Cv
1 місяць	0,15	0,07	0%	0%
2 місяці	0,14	0,07	-6,6%	0%
3 місяці	0,14	0,05	-6,6%	-28,5%
4 місяці	0,09	0,03	-40%	-57,1%
5 місяців	0,05	0,03	-66,6%	-57,1%
6 місяців	0,03	0,02	-80%	-71,4%

$$\Delta P = \frac{P_i}{P_{\text{тестування}}} \cdot 100\% - 100\% \qquad \Delta Cv = \frac{Cv_i}{Cv_{\text{тестування}}} \cdot 100\% - 100\%$$

Фактичний аналіз прогнозу

- **Показники A/B тестування:**

  Популярність: 0,06

  Конверсія: 0,09

Прогноз часу просування: 3 місяці

Авдиторія: від 30 до 55

- **Показники завершеного просування за прогнозом:**

Час просування	Популярність (P)	Конверсія (Cv)	Δ P	Δ Cv
1 місяць	0,06	0,09	0%	0%
2 місяці	0,07	0,07	+16,6%	-22,2%
3 місяці	0,06	0,07	0%	-22,2%
4 місяці	0,05	0,05	-16,6%	-44,4%
5 місяців	0,05	0,02	-16,6%	-77,7%
6 місяців	0,03	0,01	-50%	-88,8%

$$\Delta P = \frac{P_i}{P_{\text{тестування}}} \cdot 100\% - 100\% \qquad \Delta Cv = \frac{Cv_i}{Cv_{\text{тестування}}} \cdot 100\% - 100\%$$

Висновки

- **Проведено дослідження сучасної ситуації у сфері інтернет-маркетингу, виконано аналіз доступних платформ та їх інструментів**
- **Розглянуто алгоритми безумовної та умовної оптимізації та їх модифікації. Описані їх переваги та недоліки.**
- **Обґрунтовано використання обраних засобів розробки**
- **Створено програмний продукт для розробки стратегії рекламного просування у сфері інтернет-маркетингу**

Недоліки та подальші дослідження

- Інтеграція в інструменти Facebook
- Розширення набору вхідних показників
- Розширення прогнозованої вибірки
- Розробка «індивідуальних» моделей
- Використання нейромереж

Дякую за увагу!

ДОДАТОК Б

BestPromotion.cs

```
using Analytics.Mathematics;
using System;
using System.Collections.Generic;
namespace Analytics.Classes
{
    public class BestPromotion
    {
        /// <summary>
        /// Популярність товару за категоріями
        /// </summary>
        private List<CustomerStatistics> _goodsPopularity;
        /// <summary>
        /// Канали просування
        /// </summary>
        private List<Channel> _channels;
        /// <summary>
        /// Бюджет кампанії
        /// </summary>
        private double _budget;
        /// <summary>
        /// Обмеження по часу
        /// </summary>
        private double _time;
        public BestPromotion(List<CustomerStatistics> goodsPopulatity,
List<Channel> channels, double time, double budget)
```

```
        {
            _goodsPopularity = new List<CustomerStatistics>(goodsPopulatity);
            _channels = new List<Channel>(channels);
            _time = time > AuxiliaryFunctions.MaxTime ?
AuxiliaryFunctions.MaxTime : time;
            _budget = budget;
        }
        /// <summary>
        /// Конструктор з загальною агресією для всіх каналів
        /// </summary>
        public BestPromotion(List<string> popularityPaths, List<string>
channelPaths, double time, double budget, Func<double, double> agression)
        {
            _goodsPopularity = new List<CustomerStatistics>();
            foreach (var path in popularityPaths)
            {
                _goodsPopularity.Add(new CustomerStatistics(path));
            }
            _channels = new List<Channel>();
            foreach (var path in channelPaths)
            {
                _channels.Add(new Channel(path, agression));
            }
            _time = time > AuxiliaryFunctions.MaxTime ?
AuxiliaryFunctions.MaxTime:time;
            _budget = budget;
        }
        private double _targetFunction(Vector x)
        {
            var result = 0.0;
```

```
    var i_max = _channels.Count;
    var index = 0;
    for (var i = 0; i < _channels.Count; i++)
    {
        var j_max = _channels[i].Count;
        for (var j = 0; j < j_max; j++)
        {
            var k_max = _channels[i].CategoryCount(j);
            for (var k = 0; k < k_max; k++)
            {
                //Лінійне представленя трьохвимірного класу
                result += _channels[i].Penetration[j][k].Value *
                _channels[i].Conversion[j][k].Value *
                _goodsPopularity[j][k].Value *
                x[index] *
                _channels[i].Agression(x[index])
                ;
                index++;
            }
        }
    }
    return -result;
}
private Vector _targetDerivative(Vector x)
{
    var result = new List<double>();
    var i_max = _channels.Count;
    var index = 0;
    for (var i = 0; i < _channels.Count; i++)
    {
```

```
        var j_max = _channels[i].Count;
        for (var j = 0; j < j_max; j++)
        {
            var k_max = _channels[i].CategoryCount(j);
            for (var k = 0; k < k_max; k++)
            {
                var coef = _channels[i].Penetration[j][k].Value *
                _channels[i].Conversion[j][k].Value *
                _goodsPopularity[j][k].Value;
                result.Add(
                coef * (
                _channels[i].Agression(x[index]) +
                x[index] *
                Optimization.NumericDerivative(_channels[i].Agression, x[index])
                )
                );
                index++;
            }
        }
    }
    return new Vector(result);
}
/// <summary>
/// Формування проекції на допустиму множину
/// </summary>
/// <returns></returns>
private Projection _projector()
{
    var dimension = 0; //Кількість координат розвернутого масиву
    var p0Elements = new List<double>();
```

```
var pList = new List<Vector>();
var bList = new List<double>();
var i_max = _channels.Count;
for (var i = 0; i < _channels.Count; i++)
{
   var j_max = _channels[i].Count;
   for (var j = 0; j < j_max; j++)
   {
      var k_max = _channels[i].CategoryCount(j);
      for (var k = 0; k < k_max; k++)
      {
         p0Elements.Add(_channels[i].Cost * _channels[i]
         .Penetration[j][k].Value);
         dimension++;
      }
   }
}
pList.Add(new Vector(p0Elements));
bList.Add(_budget);
pList.Add((-1) * pList[0]);
bList.Add(-1.0);
var index = 0;
for (var i = 0; i < _channels.Count; i++)
{
   var j_max = _channels[i].Count;
   for (var j = 0; j < j_max; j++)
   {
      var k_max = _channels[i].CategoryCount(j);
      for (var k = 0; k < k_max; k++)
      {
```

```
                var p_i = new Vector(dimension);
                p_i[index] = 1.0;
                //Умова (p_i,t)<T (воно ж T_ijk<T)
                pList.Add(p_i);
                bList.Add(_time);
                //Умова (p_i,t)>0 (воно ж T_ijk>0)
                pList.Add((-1) * p_i);
                bList.Add(-1.0);
                index++;
            }
        }
    }
    return new Projection(dimension, pList, bList);
}
public List<double> BestStrategy()
{
    var projector = _projector();
    var result = Optimization.Optimum(projector.Dimension, _targetFunction,
_targetDerivative, projector.Project);
    return result.ToList();
}
public double Time
{
    get => _time;
    set => _time = value;
}
public double Budget
{
    get => _budget;
    set => _budget = value;
```

```
        }
    }
     }
```

Channel.cs

```
using System;
using System.Collections.Generic;
namespace Analytics.Classes
{
    public class Channel : TextDataClass
    {
        private string _name;
        private ChannelsType _type;
        private double _cost;
        private List<CustomerStatistics> _penetration;
        private List<CustomerStatistics> _conversion;
        public Func<double, double> Agression;
        public Channel(
        string name,
        ChannelsType type,
        double cost,
        List<CustomerStatistics> penetration,
        List<CustomerStatistics> conversion,
        Func<double, double> agression
        )
        {
            _name = name;
            _type = type;
            _cost = cost;
```

```
            _penetration = new List<CustomerStatistics>(penetration);
            _conversion = new List<CustomerStatistics>(conversion);
            Agression = agression;
        }
        /// <summary>
        /// Конструктор завантаження з файлу
        /// </summary>
        public Channel(
        string baseDataFilePath,
        List<string> penetrationDataFilePaths,
        List<string> conversionDataFilePaths,
        Func<double, double> agression
        )
        {
            var data = ReadFromFile(baseDataFilePath);
            _name = data.Count > 0 ? data[0] : "";
            _type = (ChannelsType)Enum.Parse(typeof(ChannelsType), data.Count > 1
? data[1] : "");
            _cost = Convert.ToDouble(data.Count > 2 ? data[2] : "");
            _penetration = new List<CustomerStatistics>();
            foreach (var penetrationDataFilePath in penetrationDataFilePaths)
            {
                _penetration.Add(new CustomerStatistics(penetrationDataFilePath));
            }
            _conversion = new List<CustomerStatistics>();
            foreach (var conversionDataFilePath in conversionDataFilePaths)
            {
                _conversion.Add(new CustomerStatistics(conversionDataFilePath
                ));
            }
```

```
        Agression = agression;
    }
    public Channel(
    string filepath,
    Func<double, double> agression
    )
    {
        var data = ReadFromFile(filepath);
        _name = data.Count > 0 ? data[0] : "";
        _type = (ChannelsType)Enum.Parse(typeof(ChannelsType), data.Count
        > 1 ? data[1] : "");
        _cost = Convert.ToDouble(data.Count > 2 ? data[2] : "");
        _penetration = new List<CustomerStatistics>();
        _conversion = new List<CustomerStatistics>();
        Agression = agression;
        var newCategory = false;
        var categoryName = "";
        var categoryElements = new List<string>();
        var categoryPenetration = new List<double>();
        var categoryConversion = new List<double>();
        var k = 0;
        for (var i = 3; i < data.Count; i++)
        {
            if (data[i] == "<new Category>")
            {
                if (categoryName != "")
                {
                    _penetration.Add(new CustomerStatistics(categoryName,
categoryElements, categoryPenetration));
```

```
                _conversion.Add(new CustomerStatistics(categoryName,
categoryElements, categoryConversion));
            }
            newCategory = true;
            categoryElements = new List<string>();
            categoryPenetration = new List<double>();
            categoryConversion = new List<double>();
            categoryName = "";
            continue;
        }
        if (newCategory)
        {
            categoryName = data[i];
            newCategory = false;
            k = 0;
            continue;
        }
        if (k % 3 == 0)
        {
            categoryElements.Add(data[i]);
            k++;
            continue;
        }
        if (k % 3 == 1)
        {
            categoryPenetration.Add(Convert.ToDouble(data[i]));
            k++;
            continue;
        }
        if (k % 3 == 2)
```

```
            {
                categoryConversion.Add(Convert.ToDouble(data[i]));
                k++;
                continue;
            }
        }
        _penetration.Add(new CustomerStatistics(categoryName,
categoryElements, categoryPenetration));
        _conversion.Add(new CustomerStatistics(categoryName,
categoryElements, categoryConversion));
    }
    /// <summary>
    /// Запись інформації у файл
    /// </summary>
    public override void SaveToFile(string filepath)
    {
        var baseData = new List<string>
{
_name,
_type.ToString(),
_cost.ToString()
};
        for (var i = 0; i < _penetration.Count; i++)
        {
            baseData.Add("<new Category>");
            baseData.Add(_penetration[i].Name);
            var countOfElements = _penetration[i].Count;
            for (var j = 0; j < countOfElements; j++)
            {
                baseData.Add(_penetration[i][j].Element);
```

```
                baseData.Add(_penetration[i][j].Value.ToString());
                baseData.Add(_conversion[i][j].Value.ToString());
            }
        }
        WriteToFile(filepath, baseData);
    }
    public string Name { get => _name; }
    public ChannelsType Type { get => _type; }
    public double Cost { get => _cost; }
    public List<CustomerStatistics> Penetration { get => _penetration; }
    public List<CustomerStatistics> Conversion { get => _conversion; }
    public int Count { get => _conversion.Count; }
    public int CategoryCount(int i) => _conversion[i].Count;
    public int TotalElementsCount
    {
        get
        {
            var result = 0;
            for (var i = 0; i < _conversion.Count; i++)
            {
                result += _conversion[i].Count;
            }
            return result;
        }
    }
    public string CategoryName(int i) => _conversion[i].Name;
}
 }
```

ChannelsType.cs

```
namespace Analytics
{
   public enum ChannelsType
   {
      Offline,
      Internet,
      TV,
      Radio
   }
}
```

CustomerCategory.cs

```
using System.Collections.Generic;
using System.Linq;
namespace Analytics.Classes
{
   /// <summary>
   /// Категория клієнтів
   /// </summary>
   public class CustomerCategory : TextDataClass
   {
      protected string _name;
      protected List<string> _elements;
      public CustomerCategory()
      {
         _name = "";
         _elements = new List<string>();
      }
```

```
public CustomerCategory(string name, List<string> elements)
{
    _name = name;
    _elements = new List<string>(elements);
}
public CustomerCategory(string filepath)
{
    var data = ReadFromFile(filepath);
    if (data.Count == 0)
    {
        _name = "";
        _elements = new List<string>();
        return;
    }
    _name = data[0];
    _elements = new List<string>(data.Skip(1));
}
public override void SaveToFile(string filepath)
{
    var data = new List<string>(_elements);
    data.Insert(0, _name);
    WriteToFile(filepath, data);
}
public override string ToString()
{
    return _name;
}
public string Name { get => _name; }
public string this[int i] { get => _elements[i]; }
public int Count { get => _elements.Count; }
```

```
    }
     }
```

CustomerStatistic.cs

```
using System;
using System.Collections.Generic;
namespace Analytics.Classes
{
   /// <summary>
   /// Спадковий клас від категорії клієнтів, відображаючий деяки метрики по елементам категорії
/// </summary>
public class CustomerStatistics : CustomerCategory
   {
      protected List<double> _metrics;
      public CustomerStatistics(string name, List<string> elements, List<double> metrics) : base(name, elements)
      {
         _metrics = new List<double>(metrics);
      }
      public CustomerStatistics(string filepath) : base()
      {
         var data = ReadFromFile(filepath);
         _metrics = new List<double>();
         if (data.Count == 0)
         {
            return;
         }
         _name = data[0];
```

```
        var dataCount = data.Count;
        for (var i = 1; i < dataCount; i++)
        {
            if (i % 2 == 1)
            {
                _elements.Add(data[i]);
                continue;
            }
            _metrics.Add(Convert.ToDouble(data[i]));
        }
        if (dataCount % 2 == 0) _metrics.Add(0.0);
    }
    public override void SaveToFile(string filepath)
    {
        var data = new List<string>();
        data.Add(_name);
        var elementsCount = _elements.Count;
        for (var i = 0; i < elementsCount; i++)
        {
            data.Add(_elements[i]);
            data.Add(_metrics[i].ToString());
        }
        WriteToFile(filepath, data);
    }
    public new Matching this[int i]
    {
        get => new Matching(_elements[i], _metrics[i]);
    }
}
 }
```

Matching.cs

```
namespace Analytics
{
   /// <summary>
   /// Структура, для деякого рядка - числове значення
   /// </summary>
   public struct Matching
   {
      public string Element;
      public double Value;
      public Matching(string element, double value)
      {
         Element = element;
         Value = value;
      }
   }
    }
```

TextDataClass.cs

```
using System;
using System.Collections.Generic;
using System.IO;
namespace Analytics.Classes
{
   public abstract class TextDataClass
   {
      public List<string> ReadFromFile(string filepath)
```

```
{
    var result = new List<string>();
    try
    {
        using (var streamReader = new StreamReader(filepath))
        {
            string line;
            while ((line = streamReader.ReadLine()) != null)
            {
                result.Add(line);
            }
        }
    }
    catch (Exception e)
    {
        result.Clear();
        result.Add(e.Message);
    }
    return result;
}
public string WriteToFile(string filepath, List<string> data)
{
    try
    {
        using (var streamWriter = new StreamWriter(filepath))
        {
            foreach (var line in data)
            {
                streamWriter.WriteLine(line);
            }
```

```
            }
        }
        catch (Exception e)
        {
            return e.Message;
        }
        return "Success";
    }
    /// <summary>
    /// Абстрактний метод запису даних в файл. Реалізується в кожномукласі окремо.
   /// </summary>
public abstract void SaveToFile(string filepath);
   }
    }
```

AuxiliaryFunctions.cs

```
using System;
namespace Analytics.Mathematics
{
   public static class AuxiliaryFunctions
   {
      public const double Eps = 0.00001;
      public const long MaxSteps = 100000;
      public const double MaxTime = 10;
      public static double testAgression(double x) => MaxTime * MaxTime - x
      * x;
   }
    }
```

Decomposition.cs

```
using System;
using System.Collections.Generic;
namespace Analytics.Mathematics
{
   public static class Decomposition
   {
      /// <summary>
      /// LUP розклад квадратної матриці, де PA=LU
      /// </summary>
      public static Tuple<Matrix, Matrix, Matrix, int> LUP(Matrix matrix)
      {
         var dimension = matrix.LinesCount;
         var U = new Matrix(matrix.ToArray());
         var L = new Matrix(dimension, dimension);
         var P = new Matrix(dimension, dimension);
         var reversionsCount = 0;
         for (var i = 0; i < dimension; i++) L[i, i] = P[i, i] = 1;
         var maxElementIndex = 0;
         for (var i = 0; i < dimension - 1; i++)
         {
            maxElementIndex = i;
            for (var j = i; j < dimension; j++)
               maxElementIndex = (Math.Abs(U[j, i]) >
Math.Abs(U[maxElementIndex, i])) ? j : maxElementIndex;
            if (Math.Abs(U[maxElementIndex, i]) < AuxiliaryFunctions.Eps)
               continue;
            if (i != maxElementIndex)
```

```
            {
                U.LinesReversion(i, maxElementIndex);
                P.LinesReversion(i, maxElementIndex);
                reversionsCount++;
            }
            for (var j = i + 1; j < dimension; j++)
            {
                U[j, i] /= U[i, i];
                for (int k = i + 1; k < dimension; k++)
                    U[j, k] -= U[j, i] * U[i, k];
            }
        }
        for (var i = 0; i < dimension; i++)
            for (var j = 0; j < i; j++)
            {
                L[i, j] = U[i, j];
                U[i, j] = 0;
            }
        return new Tuple<Matrix, Matrix, Matrix, int>(L, U, P, reversionsCount);
    }
    /// <summary>
    /// Розклад Гауса (не використовується)
    /// </summary>
    public static Tuple<List<SimpleAction>, Matrix>
GaussianElimination(Matrix matrix, bool maxElementFinding = true, bool upper =
true)
    {
        var result = new Matrix(matrix.ToArray());
        var simpleActions = new List<SimpleAction>();
```

```
        var stepCount = (result.LinesCount > result.ColumnsCount) ?
result.ColumnsCount : result.LinesCount;
        var linesCount = result.LinesCount;
        var columnsCount = result.ColumnsCount;
        for (var i = 0; i < stepCount; i++)
        {
            var currentLineIndex = (upper) ? i : linesCount - 1 - i;
            var currentColumnIndex = (upper) ? i : columnsCount - 1 - i;
            var mainElementIndex = (upper) ? i : linesCount - 1 - i;
            for (var linesStep = i; linesStep < result.LinesCount; linesStep++)
            {
                var linesIndex = (upper) ? linesStep : linesCount - linesStep - 1;
                if (maxElementFinding) mainElementIndex =
(Math.Abs(result[linesIndex, currentColumnIndex]) >
                Math.Abs(result[mainElementIndex, currentColumnIndex])) ?
linesIndex : mainElementIndex;
                else
                if (Math.Abs(result[linesIndex, currentColumnIndex]) >
AuxiliaryFunctions.Eps)
{
            mainElementIndex = linesIndex;
            break;
        }
    }
if (Math.Abs(result[mainElementIndex, currentColumnIndex]) <
AuxiliaryFunctions.Eps)
continue;
if (currentLineIndex != mainElementIndex)
{
result.LinesReversion(currentLineIndex, mainElementIndex)
```

```
;
simpleActions.Add(new SimpleAction(SimpleActionType.LinesReversion,
currentLineIndex, mainElementIndex));
}
   var coefficient1 = (1 / result[currentLineIndex, currentColumnIndex]);
for (var columnsStep = i; columnsStep<columnsCount; columnsStep++)
{
var columnsIndex = (upper) ? columnsStep : columnsCount -
columnsStep - 1;
   result[currentLineIndex, columnsIndex] *= coefficient1;
}
simpleActions.Add(new
SimpleAction(SimpleActionType.LineCoef,currentLineIndex, coefficient1));
for (var linesStep = i + 1; linesStep<result.LinesCount; linesStep++)
{
var linesIndex = (upper) ? linesStep : linesCount - linesStep - 1;
var coefficient2 = -
result[linesIndex, currentColumnIndex];
for (var columnsStep = i; columnsStep<result.ColumnsCount; columnsStep++)
{
var columnsIndex = (upper) ? columnsStep : columnsCount - columnsStep - 1;
result[linesIndex, columnsIndex] += coefficient2* result[currentLineIndex,
columnsIndex];
}
simpleActions.Add(
new SimpleAction(SimpleActionType.LinesSumCoef, linesIndex,
currentLineIndex, coefficient2)
);
}
}
```

```
return new Tuple<List<SimpleAction>, Matrix>(simpleActions, result);
}
}
    }
```

LinearSystem.cs

```
namespace Analytics.Mathematics
{
   public static class LinearSystem
   {
      /// <summary>
      /// Вирішення матричних рівнянь вигляду: AX=B
      /// </summary>
      public static Matrix LeftLUP(Matrix leftSide, Matrix rightSide)
      {
         var leftSideDimension = leftSide.LinesCount;
         var lupResult = Decomposition.LUP(leftSide);
         var _reversedRightSide = lupResult.Item3 * rightSide;
         var result = new Matrix(rightSide.LinesCount, rightSide.ColumnsCount);
         for (var j = 0; j < _reversedRightSide.ColumnsCount; j++)
         {
            for (int i = 0; i < leftSideDimension; i++)
            {
               var s = 0.0;
               if (i != 0)
                  for (var k = 0; k < i; k++) s += lupResult.Item1[i, k
                  ] * result[k, j];
               result[i, j] = _reversedRightSide[i, j] - s;
            }
```

```
            }
            _reversedRightSide = result;
            for (var j = 0; j < _reversedRightSide.ColumnsCount; j++)
            {
                for (var i = leftSideDimension - 1; i >= 0; i--)
                {
                    var s = 0.0;
                    if (i != leftSideDimension - 1)
                        for (var k = i + 1; k < leftSideDimension; k++) s +=
                        lupResult.Item2[i, k] * result[k, j];
                    result[i, j] = (1 / lupResult.Item2[i, i]) * (_reversedRightSide[i, j] - s);
        }
    }
return result;
}
public static Matrix RightLUP(Matrix leftSide, Matrix rightSide)
{
    var leftSideDimension = leftSide.LinesCount;
    var lupResult = Decomposition.LUP(leftSide);
    var tempRightSide = rightSide;
    var result = new Matrix(rightSide.LinesCount, rightSide.ColumnsCount);
    for (var j = 0; j < tempRightSide.ColumnsCount; j++)
    {
        for (int i = 0; i < tempRightSide.LinesCount; i++)
        {
            var s = 0.0;
            if (j != 0)
                for (var k = 0; k < j; k++) s += lupResult.Item2[k, j
                ] * result[i, k];
            result[i, j] = (tempRightSide[i, j] - s) / lupResult.Item2[j, j];
```

```
        }
    }
    tempRightSide = result;
    for (var j = tempRightSide.ColumnsCount - 1; j >= 0; j--)
    {
        for (var i = 0; i < tempRightSide.LinesCount; i++)
        {
            var s = 0.0;
            if (j != tempRightSide.ColumnsCount - 1)
                for (var k = j + 1; k < tempRightSide.ColumnsCount; k
                ++) s += lupResult.Item1[k, j] * result[i, k];
            result[i, j] = tempRightSide[i, j] - s;
        }
    }
    return result * lupResult.Item3;
}
public static Matrix LUP(Matrix leftSideLeft, Matrix leftSideRight, Matrix
rightSide)
=> RightLUP(leftSideRight, LeftLUP(leftSideLeft, rightSide));
public static Matrix Inverse(Matrix matrix)
{
    var I = new Matrix(matrix.LinesCount, matrix.ColumnsCount);
    for (var i = 0; i < matrix.LinesCount; i++) I[i, i] = 1.0;
    return LeftLUP(matrix, I);
}
}
    }
```

Matrix.cs

```
using System.Collections.Generic;
namespace Analytics.Mathematics
{
    public class Matrix
    {
        private int _linesCount;
        private int _columnsCount;
        private double[,] _array;
        public Matrix(int linesCount, int columnsCount)
        {
            _linesCount = linesCount;
            _columnsCount = columnsCount;
            _array = new double[_linesCount, _columnsCount];
            for (var i = 0; i < _linesCount; i++)
                for (var j = 0; j < _columnsCount; j++) _array[i, j] = 0;
        }
        public Matrix(List<Vector> vectors, bool asLines = true)
        {
            if (asLines)
            {
                _linesCount = vectors.Count;
                _columnsCount = _linesCount > 0 ? vectors[0].Dimension : 0;
                _array = new double[_linesCount, _columnsCount];
                for (var i = 0; i < _linesCount; i++)
                    for (var j = 0; j < _columnsCount; j++) _array[i, j] = vectors[i][j];
                return;
            }
            _columnsCount = vectors.Count;
            _linesCount = _columnsCount > 0 ? vectors[0].Dimension : 0;
            _array = new double[_linesCount, _columnsCount];
```

```
    for (var i = 0; i < _linesCount; i++)
        for (var j = 0; j < _columnsCount; j++) _array[i, j] = vectors[j][i];
}
public Matrix(double[,] array)
{
    _linesCount = array.GetLength(0);
    _columnsCount = array.GetLength(1);
    _array = new double[_linesCount, _columnsCount];
    for (var i = 0; i < _linesCount; i++)
        for (var j = 0; j < _columnsCount; j++) _array[i, j] = array[
        i, j];
}
public override string ToString()
{
    var result = "";
    for (var i = 0; i < _linesCount; i++)
        for (var j = 0; j < _columnsCount; j++)
            result += _array[i, j].ToString() + ((j < _columnsCount -
            1) ? " " : ((i < _linesCount - 1) ? "\r\n" : ""));
    return result;
}
public int LinesCount { get => _linesCount; }
public int ColumnsCount { get => _columnsCount; }
public double this[int lineIndex, int columnIndex]
{
    get => _array[lineIndex, columnIndex];
    set => _array[lineIndex, columnIndex] = value;
}
public List<Vector> Lines
{
```

```
    get
    {
        var result = new List<Vector>();
        for (var i = 0; i < _linesCount; i++)
        {
            var vector = new Vector(_columnsCount);
            for (var j = 0; j < _columnsCount; j++) vector[j] = _array[i, j];
            result.Add(vector);
        }
        return result;
    }
}
public List<Vector> Columns
{
    get
    {
        var result = new List<Vector>();
        for (var j = 0; j < _columnsCount; j++)
        {
            var vector = new Vector(_linesCount);
            for (var i = 0; i < _linesCount; i++) vector[i] = _array[
            i, j];
            result.Add(vector);
        }
        return result;
    }
}
public void VectorToLine(int lineIndex, Vector vector)
{
    for (var j = 0; j < _columnsCount; j++)
```

```
            _array[lineIndex, j] = (j < vector.Dimension) ? vector[j] : 0
            ;
        }
        public void VectorToColumn(int columnIndex, Vector vector)
        {
            for (var i = 0; i < _linesCount; i++)
            _array[i, columnIndex] = (i < vector.Dimension) ? vector[i] :
            0;
        }
        public void LinesReversion(int lineIndex1, int lineIndex2)
        {
            var temp = Lines[lineIndex1];
            VectorToLine(lineIndex1, Lines[lineIndex2]);
            VectorToLine(lineIndex2, temp);
        }
        public void ColumnsReversion(int columnIndex1, int columnIndex2)
        {
            var temp = Columns[columnIndex1];
            VectorToColumn(columnIndex1, Columns[columnIndex2]);
            VectorToColumn(columnIndex2, temp);
        }
        public static Matrix operator +(Matrix matrix1, Matrix matrix2)
        {
            var resultColumnsCount = (matrix1.ColumnsCount >=
matrix2.ColumnsCount) ?
matrix1.ColumnsCount : matrix2.ColumnsCount;
            var resultLinesCount = (matrix1.LinesCount >= matrix2.LinesCount)
            ?
            matrix1.LinesCount : matrix2.LinesCount;
            var result = new Matrix(resultLinesCount, resultColumnsCount);
```

```
            for (var i = 0; i < resultLinesCount; i++)
                for (var j = 0; j < resultColumnsCount; j++)
                    result[i, j] = ((i < matrix1.LinesCount && j < matrix1.ColumnsCount)
? matrix1[i, j] : 0) + ((i < matrix2.LinesCount && j < matrix2.ColumnsCount) ?
matrix2[i, j] : 0);
            return result;
        }
        public static Matrix operator -(Matrix matrix1, Matrix matrix2)
        {
            var resultColumnsCount = (matrix1.ColumnsCount >=
matrix2.ColumnsCount) ?
matrix1.ColumnsCount : matrix2.ColumnsCount;
            var resultLinesCount = (matrix1.LinesCount >= matrix2.LinesCount)
            ?
            matrix1.LinesCount : matrix2.LinesCount;
            var result = new Matrix(resultLinesCount, resultColumnsCount);
            for (var i = 0; i < resultLinesCount; i++)
                for (var j = 0; j < resultColumnsCount; j++)
                    result[i, j] = ((i < matrix1.LinesCount && j < matrix1.ColumnsCount)
? matrix1[i, j] : 0) - ((i < matrix2.LinesCount && j < matrix2.ColumnsCount) ?
matrix2[i, j] : 0);
            return result;
        }
        public static Matrix operator *(double coefficient, Matrix matrix)
        {
            var result = new Matrix(matrix._linesCount, matrix._columnsCount)
            ;
            for (var i = 0; i < result._linesCount; i++)
                for (var j = 0; j < result._columnsCount; j++) result[i, j] =
                coefficient * matrix[i, j];
```

```
    return result;
}
public static Matrix operator *(Matrix matrix1, Matrix matrix2)
{
    var resultColumnsCount = matrix2.ColumnsCount;
    var resultLinesCount = matrix1.LinesCount;
    var result = new Matrix(resultLinesCount, resultColumnsCount);
    for (var i = 0; i < resultLinesCount; i++)
        for (var j = 0; j < resultColumnsCount; j++)
            result[i, j] = matrix1.Lines[i] * matrix2.Columns[j];
    return result;
}
public static Vector operator *(Matrix matrix, Vector vector)
{
    var resultDimension = matrix.LinesCount;
    var result = new Vector(resultDimension);
    for (var i = 0; i < resultDimension; i++)
        result[i] = matrix.Lines[i] * vector;
    return result;
}
public Matrix Transposed()
{
    var result = new Matrix(_columnsCount, _linesCount);
    for (var i = 0; i < _linesCount; i++)
    {
        result.VectorToColumn(i, Lines[i]);
    }
    return result;
}
public double Trace()
```

```
        {
            var mainDiagonalLength = (_linesCount > _columnsCount) ?
_columnsCount : _linesCount;
           var result = 0.0;
            for (int i = 0; i < mainDiagonalLength; i++) result += _array[i,
            i];
            return result;
        }
        public double[,] ToArray()
        {
            var result = new double[_linesCount, _columnsCount];
            for (var i = 0; i < _linesCount; i++)
                for (var j = 0; j < _columnsCount; j++) result[i, j] = _array
                [i, j];
            return result;
        }
    }
     }
```

Optimization.cs

```
using System;
namespace Analytics.Mathematics
{
    public static class Optimization
    {
        /// <summary>
        /// Числова похідна
        /// </summary>
        /// <param name="f">Функція</param>
```

```
        /// <param name="x">Числовий аргумент</param>
        /// <returns></returns>
        public static double NumericDerivative(Func<double, double> f, double
        x)
        {
            var h = 1 / AuxiliaryFunctions.Eps;
            return 0.5 * AuxiliaryFunctions.Eps * (f(x + h) - f(x - h));
        }
        /// <summary>
        /// Метод найшвидшого спуску для підбору розміру кроку з
використанням методу золотого перетину
    /// </summary>
/// <param name="f">Цільова функція</param>
/// <param name="h">Крок</param>
/// <param name="x">Точка</param>
/// <returns></returns>
public static double FastestFall(Func<Vector, double> f, Vector h, Vector x)
        {
            double a = 0; double b = 1;
            double phi = (Math.Sqrt(5) + 1) / 2;
            double x1, x2;
            double y1, y2;
            int k = 0;
            do
            {
                x1 = b - (b - a) / phi;
                x2 = a + (b - a) / phi;
                y1 = f(x + x1 * h);
                y2 = f(x + x2 * h);
                if (y1 >= y2) a = x1;
```

```
            else b = x2;
            k++;
        } while (Math.Abs(b - a) >= AuxiliaryFunctions.Eps && k <
AuxiliaryFunctions.MaxSteps);
        return (a + b) / 2;
    }
    /// <summary>
    /// Градієнтний метод оптимізації
    /// </summary>
    public static Vector Optimum(
    int dimension,
    Func<Vector, double> f,
    Func<Vector, Vector> df,
    Func<Vector, Vector> projection
    )
    {
        var x_cur = new Vector(dimension);
        var x_next = new Vector(dimension);
        for (var i = 0; i < x_next.Dimension; i++) x_next[i] = (1 +
AuxiliaryFunctions.MaxTime) / 2;
        var h_cur = new Vector(dimension);
        double a_k = 0;
        int k = 0;
        do
        {
            k++;
            x_cur = new Vector(x_next.ToList());
            h_cur = (-1) * df(x_cur);
            a_k = FastestFall(f, h_cur, x_cur);
            x_next = projection(x_cur + a_k * h_cur);
```

```
            k++;
        } while (((x_next - x_cur) >= AuxiliaryFunctions.Eps) && k <
AuxiliaryFunctions.MaxSteps);
        return x_cur;
      }
   }
    }
```

Projection.cs

```
using System.Collections.Generic;
namespace Analytics.Mathematics
{
   /// <summary>
   /// Пошук проекції на поліедр: (x,pi)<bi
   /// </summary>
   public class Projection
   {
      private int _dimension;
      private Matrix _a;
      private Vector _b;
      private Matrix _aTsqra;
      private List<Vector> _p;
      public Projection(int dimension, List<Vector> p, List<double> b)
      {
         _dimension = dimension;
         _b = new Vector(b);
         _a = new Matrix(p);
         var _aT = _a.Transposed();
         _aTsqra = _aT * LinearSystem.Inverse(_a * _aT);
```

```
    _p = new List<Vector>(p);
}
public Vector Project(Vector x)
{
    var x_cur = new Vector(x.ToList());
    for (var i = 0; i < _b.Dimension; i++)
    {
        if (_p[i] * x_cur <= _b[i])
        {
            continue;
        }
        x_cur = x_cur + (_b[i] - _p[i] * x_cur) * (1.0 / (_p[i] * _p[i
        ])) * _p[i];
    }
    return x_cur;
}
/// <summary>
/// Функція-проектор
/// </summary>
public Vector ProjectNya(Vector x)
{
    var ax = _a * x;
    var belongs = true;
    for (var i = 0; i < x.Dimension; i++)
        if (_b[i] - ax[i] < -AuxiliaryFunctions.Eps)
        {
            belongs = false;
            break;
        }
    if (belongs) return new Vector(x.ToList());
```

```
            return x - _aTsqra * (_a * x - _b);
        }
        public int Dimension { get => _dimension; }
    }
     }
```

SimpleAction.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Analytics.Mathematics
{
    public enum SimpleActionType
    {
        None,
        LinesReversion,
        LineCoef,
        LinesSum,
        LinesSumCoef,
        ColumnsReversion,
        ColumnCoef,
        ColumnsSum,
        ColumnsSumCoef
    }
    public class SimpleAction
    {
        private SimpleActionType _type;
```

```
        private int _firstVector;
        private int _secondVector;
        private double _coefficient;
        public SimpleActionType Type { get => _type; }
        public int FirstVector { get => _firstVector; }
        public int SecondVector { get => _secondVector; }
        public double Coefficient { get => _coefficient; }
        public SimpleAction(SimpleActionType type, int firstVector, int
secondVector)
        {
            _type = type;
            _firstVector = firstVector;
            _secondVector = secondVector;
            _coefficient = 0;
        }
        public SimpleAction(SimpleActionType type, int firstVector, double
coefficient)
        {
            _type = type;
            _firstVector = firstVector;
            _secondVector = -1;
            _coefficient = coefficient;
        }
        public SimpleAction(SimpleActionType type, int firstVector, int
secondVector, double coefficient)
        {
            _type = type;
            _firstVector = firstVector;
            _secondVector = secondVector;
            _coefficient = coefficient;
```

```
        }
    }
     }
```

Vector.cs

```
using System;
using System.Collections.Generic;
namespace Analytics.Mathematics
{
    public class Vector
    {
        private int _dimension;
        private double[] _array;
        public Vector(int dimension)
        {
            _dimension = dimension;
            _array = new double[_dimension];
            for (var i = 0; i < _dimension; i++) _array[i] = 0;
        }
        public Vector(List<double> preassignedCoordinates)
        {
            _dimension = preassignedCoordinates.Count;
            _array = new double[_dimension];
            for (var i = 0; i < _dimension; i++) _array[i] = preassignedCoordinates[i];
        }
        public override string ToString()
        {
            var result = "";
            for (var i = 0; i < _dimension; i++) result +=
```

```
    (_array[i].ToString() + ((i < _dimension - 1) ? " " : "")
    );
    return result;
}
public int Dimension { get => _dimension; }
public double this[int index]
{
    get => _array[index];
    set => _array[index] = value;
}
public double Max
{
    get
    {
        var result = _array[0];
        foreach (var coordinate in _array)
        {
            result = (result < coordinate) ? coordinate : result;
        }
        return result;
    }
}
public double Min
{
    get
    {
        var result = _array[0];
        foreach (var coordinate in _array)
        {
            result = (result > coordinate) ? coordinate : result;
```

```
        }
        return result;
    }
}
public List<double> ToList() => new List<double>(_array);
public void ElementsReversion(int index1, int index2)
{
    double temp = _array[index1];
    _array[index1] = _array[index2];
    _array[index2] = temp;
}
public static Vector operator +(Vector vector1, Vector vector2)
{
    var resultDimension = (vector1.Dimension >= vector2.Dimension) ?
    vector1.Dimension : vector2.Dimension;
    var result = new Vector(resultDimension);
    for (var i = 0; i < resultDimension; i++)
        result[i] = ((i < vector1.Dimension) ? vector1[i] : 0) +
        ((i < vector2.Dimension) ? vector2[i] : 0);
    return result;
}
public static Vector operator -(Vector vector1, Vector vector2)
{
    var resultDimension = (vector1.Dimension >= vector2.Dimension) ?
    vector1.Dimension : vector2.Dimension;
    var result = new Vector(resultDimension);
    for (var i = 0; i < resultDimension; i++)
        result[i] = ((i < vector1.Dimension) ? vector1[i] : 0) -
        ((i < vector2.Dimension) ? vector2[i] : 0);
    return result;
```

```
    }
    public static Vector operator *(double coefficient, Vector vector)
    {
        var result = new Vector(vector.ToList());
        for (var i = 0; i < result.Dimension; i++) result[i] *= coefficient;
        return result;
    }
    public static double operator *(Vector vector1, Vector vector2)
    {
        var resultDimension = (vector1.Dimension <= vector2.Dimension) ?
        vector1.Dimension : vector2.Dimension;
        var result = 0.0;
        for (var i = 0; i < resultDimension; i++)
            result += vector1[i] * vector2[i];
        return result;
    }
    public double Norm { get => Math.Sqrt(this * this); }
    public bool IfZero { get => Norm < AuxiliaryFunctions.Eps; }
    public override bool Equals(object obj)
    {
        if (obj == null) return false;
        var temp = obj as Vector;
        if ((object)temp == null) return false;
        return (temp - this).IfZero;
    }
    public override int GetHashCode() => _dimension.GetHashCode() ^ _array
    .GetHashCode();
    public static bool operator ==(Vector vector1, Vector vector2) =>
vector1.Equals(vector2);
```

```
public static bool operator !=(Vector vector1, Vector vector2) =>
!vector1.Equals(vector2);
public static bool operator >(Vector vector, double radius) => vector.
Norm > radius;
    public static bool operator <(Vector vector, double radius) => vector.
    Norm < radius;
    public static bool operator >=(Vector vector, double radius) => vector
    .Norm >= radius;
    public static bool operator <=(Vector vector, double radius) => vector
    .Norm <= radius;
    public static bool operator ==(Vector vector, double radius) => vector
    .Norm == radius;
    public static bool operator !=(Vector vector, double radius) => vector
    .Norm != radius;
    public Vector Normed => (1 / Norm) * this;
  }
  }
```

Form1.cs

```
using Analytics.Classes;
using Analytics.Mathematics;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
using System.Windows.Forms;

namespace Analytics
{
   public partial class Form1 : Form
   {
      public Color darkColor, mediumColor, lightColor;
      public Panel ioPanel, goodsPanel, channelsPanel, textPanel;
      //textPanel
      public TextBox textBox;
      //inputPanel
      public Label iPopularity, iChannels, iBudget, iTime;
      public TextBox fpPopularity, fpChannels;
      public Button fbPopularity, fbChannels;
      public Button bPopularity, bChannel;
      public TextBox tBudget, tTime;
      public Button start;
      //goodsPanel
      public Label tPopularity;
      public DataGridView dPopularity;
      //channelsPanel
      public Label tChannels;
      public DataGridView dChannels;
      //not interface
      public List<Channel> channels;
      public List<CustomerStatistics> goodsPopularity;
      public Form1()
      {
         goodsPopularity = new List<CustomerStatistics>();
         channels = new List<Channel>();
```

```
components = new Container();
AutoScaleMode = AutoScaleMode.Font;
Text = "Analytics";
MaximizeBox = false;
darkColor = Color.FromArgb(87, 87, 87);
mediumColor = Color.FromArgb(168, 168, 168);
lightColor = Color.FromArgb(192, 245, 132);
Size = new Size(1100, 800);
ClientSize = new Size(1100, 800);
AutoSize = false;
AutoScroll = false;
FormBorderStyle = FormBorderStyle.FixedSingle;
Load += new EventHandler(_formLoad);
ioPanel = new Panel()
{
   Location = new Point(0, 0),
   Size = new Size(200, 600),
   BackColor = mediumColor,
   BorderStyle = BorderStyle.FixedSingle
};
Controls.Add(ioPanel);
goodsPanel = new Panel()
{
   Location = new Point(200, 0),
   Size = new Size(300, 600),
   BackColor = mediumColor,
   BorderStyle = BorderStyle.FixedSingle
};
Controls.Add(goodsPanel);
channelsPanel = new Panel()
```

```
{
    Location = new Point(500, 0),
    Size = new Size(700, 600),
    BackColor = mediumColor,
    BorderStyle = BorderStyle.FixedSingle
};
Controls.Add(channelsPanel);
textPanel = new Panel()
{
    Location = new Point(0, 600),
    Size = new Size(1200, 200),
    BackColor = mediumColor,
    BorderStyle = BorderStyle.FixedSingle
};
Controls.Add(textPanel);
//textBox
textBox = new TextBox()
{
    Location = new Point(0, 0),
    Size = new Size(textPanel.Width, textPanel.Height),
    BackColor = lightColor,
    ReadOnly = true,
    ForeColor = darkColor,
    Multiline = true
};
textPanel.Controls.Add(textBox);
//Input panel
iPopularity = new Label()
{
    Location = new Point(0, 0),
```

```
    AutoSize = false,
    TextAlign = ContentAlignment.MiddleCenter,
    Font = new Font(new FontFamily("Times New Roman"), 10),
    Width = ioPanel.Width,
    Height = 25,
    BackColor = darkColor,
    ForeColor = lightColor,
    Text = "Популярность за категоріями"
};
ioPanel.Controls.Add(iPopularity);
fpPopularity = new TextBox()
{
    Location = new Point(0, iPopularity.Bottom + 10),
    Width = ioPanel.Width - 25,
    Height = 25,
    Font = new Font(new FontFamily("Times New Roman"), 10),
    BackColor = lightColor,
    ForeColor = darkColor,
    ReadOnly = true
};
ioPanel.Controls.Add(fpPopularity);
fbPopularity = new Button()
{
    Location = new Point(fpPopularity.Right, fpPopularity.Top),
    Width = 25,
    Height = fpPopularity.Height,
    BackColor = lightColor,
    ForeColor = darkColor,
    Text = "..."
};
```

```
fbPopularity.Click += new EventHandler(_openPopularityDialog);
ioPanel.Controls.Add(fbPopularity);
bPopularity = new Button()
{
   Location = new Point(50, fpPopularity.Bottom + 10),
   Width = 100,
   Height = 50,
   BackColor = lightColor,
   ForeColor = darkColor,
   Text = "Додати категорію"
};
bPopularity.Click += new EventHandler(_addPopularity);
ioPanel.Controls.Add(bPopularity);
//
iChannels = new Label()
{
   Location = new Point(0, bPopularity.Bottom + 25),
   AutoSize = false,
   TextAlign = ContentAlignment.MiddleCenter,
   Font = new Font(new FontFamily("Times New Roman"), 10),
   Width = ioPanel.Width,
   Height = 25,
   BackColor = darkColor,
   ForeColor = lightColor,
   Text = "Канали"
};
ioPanel.Controls.Add(iChannels);
fpChannels = new TextBox()
{
   Location = new Point(0, iChannels.Bottom + 10),
```

```
    Width = ioPanel.Width - 25,
    Height = 25,
    Font = new Font(new FontFamily("Times New Roman"), 10),
    BackColor = lightColor,
    ForeColor = darkColor,
    ReadOnly = true
};
ioPanel.Controls.Add(fpChannels);
fbChannels = new Button()
{
    Location = new Point(fpChannels.Right, fpChannels.Top),
    Width = 25,
    Height = fpChannels.Height,
    BackColor = lightColor,
    ForeColor = darkColor,
    Text = "..."
};
fbChannels.Click += new EventHandler(_openChannelsDialog);
ioPanel.Controls.Add(fbChannels);
bChannel = new Button()
{
    Location = new Point(50, fpChannels.Bottom + 10),
    Width = 100,
    Height = 50,
    BackColor = lightColor,
    ForeColor = darkColor,
    Text = "Додати канал"
};
bChannel.Click += new EventHandler(_addChannel);
ioPanel.Controls.Add(bChannel);
```

```
//
iBudget = new Label()
{
   Location = new Point(0, bChannel.Bottom + 25),
   AutoSize = false,
   TextAlign = ContentAlignment.MiddleCenter,
   Font = new Font(new FontFamily("Times New Roman"), 10),
   Width = ioPanel.Width,
   Height = 25,
   BackColor = darkColor,
   ForeColor = lightColor,
   Text = "Бюджет"
};
ioPanel.Controls.Add(iBudget);
tBudget = new TextBox()
{
   Location = new Point(0, iBudget.Bottom + 10),
   Width = ioPanel.Width,
   Height = 25,
   Font = new Font(new FontFamily("Times New Roman"), 10),
   BackColor = lightColor,
   ForeColor = darkColor,
   TextAlign = HorizontalAlignment.Center,
   Text = "1000"
};
ioPanel.Controls.Add(tBudget);
//
iTime = new Label()
{
   Location = new Point(0, tBudget.Bottom + 25),
```

```
    AutoSize = false,
    TextAlign = ContentAlignment.MiddleCenter,
    Font = new Font(new FontFamily("Times New Roman"), 10),
    Width = ioPanel.Width,
    Height = 25,
    BackColor = darkColor,
    ForeColor = lightColor,
    Text = "Час"
};
ioPanel.Controls.Add(iTime);
tTime = new TextBox()
{
    Location = new Point(0, iTime.Bottom + 10),
    Width = ioPanel.Width,
    Height = 25,
    Font = new Font(new FontFamily("Times New Roman"), 10),
    BackColor = lightColor,
    ForeColor = darkColor,
    TextAlign = HorizontalAlignment.Center,
    Text = AuxiliaryFunctions.MaxTime.ToString()
};
ioPanel.Controls.Add(tTime);
//
start = new Button()
{
    Location = new Point(50, tTime.Bottom + 50),
    Width = 100,
    Height = 50,
    BackColor = lightColor,
    ForeColor = darkColor,
```

```
            Text = "Старт"
        };
        start.Click += new EventHandler(_start);
        ioPanel.Controls.Add(start);
        //goodsPanel
        tPopularity = new Label()
        {
            Location = new Point(0, 0),
            AutoSize = false,
            TextAlign = ContentAlignment.MiddleCenter,
            Font = new Font(new FontFamily("Times New Roman"), 10),
            Width = goodsPanel.Width,
            Height = 25,
            BackColor = darkColor,
            ForeColor = lightColor,
            Text = "Популярність товару"
        };
        goodsPanel.Controls.Add(tPopularity);
        dPopularity = new DataGridView()
        {
            Location = new Point(0, tPopularity.Bottom),
            Size = new Size(300, 575),
            BackgroundColor = lightColor,
            ColumnCount = 3,
            ReadOnly = true,
            AllowUserToResizeRows = false,
            AllowUserToResizeColumns = false,
            ColumnHeadersHeightSizeMode =
DataGridViewColumnHeadersHeightSizeMode.DisableResizing,
            RowHeadersVisible = false,
```

```
            ColumnHeadersVisible = true,
            ColumnHeadersHeight = 50,
            AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.None,
            AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.None,
        };
        dPopularity.EnableHeadersVisualStyles = false;
        dPopularity.ColumnHeadersDefaultCellStyle.ForeColor = lightColor;
        dPopularity.ColumnHeadersDefaultCellStyle.BackColor = darkColor;
        dPopularity.DefaultCellStyle.ForeColor = darkColor;
        dPopularity.DefaultCellStyle.BackColor = lightColor;
        dPopularity.Columns[0].HeaderCell.Style.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        dPopularity.Columns[1].HeaderCell.Style.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        dPopularity.Columns[2].HeaderCell.Style.Alignment =
DataGridViewContentAlignment.MiddleCenter;
        dPopularity.Columns[0].Width = 100;
        dPopularity.Columns[1].Width = 100;
        dPopularity.Columns[2].Width = 100;
        dPopularity.Columns[0].HeaderText = "Категорія";
        dPopularity.Columns[1].HeaderText = "Елемент категорії";
        dPopularity.Columns[2].HeaderText = "Популярність";
        dPopularity.Columns[0].SortMode =
DataGridViewColumnSortMode.NotSortable;
        dPopularity.Columns[1].SortMode =
DataGridViewColumnSortMode.NotSortable;
        dPopularity.Columns[2].SortMode =
DataGridViewColumnSortMode.NotSortable;
        goodsPanel.Controls.Add(dPopularity);
```

```
            //channelsPanel
            tChannels = new Label()
            {
               Location = new Point(0, 0),
               AutoSize = false,
               TextAlign = ContentAlignment.MiddleCenter,
               Font = new Font(new FontFamily("Times New Roman"), 10),
               Width = channelsPanel.Width,
               Height = 25,
               BackColor = darkColor,
               ForeColor = lightColor,
               Text = "Дані за каналами"
            };
            channelsPanel.Controls.Add(tChannels);
            dChannels = new DataGridView()
            {
               Location = new Point(0, tChannels.Bottom),
               Size = new Size(600, 575),
               BackgroundColor = lightColor,
               ColumnCount = 6,
               ReadOnly = true,
               AllowUserToResizeRows = false,
               AllowUserToResizeColumns = false,
               ColumnHeadersHeightSizeMode                                    =
DataGridViewColumnHeadersHeightSizeMode.DisableResizing,
               RowHeadersVisible = false,
               ColumnHeadersVisible = true,
               ColumnHeadersHeight = 50,
               AutoSizeColumnsMode                                            =
DataGridViewAutoSizeColumnsMode.None,
```

```
            AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.None,
        };
        dChannels.EnableHeadersVisualStyles = false;
        dChannels.ColumnHeadersDefaultCellStyle.ForeColor = lightColor;
        dChannels.ColumnHeadersDefaultCellStyle.BackColor = darkColor;
        dChannels.DefaultCellStyle.ForeColor = darkColor;
        dChannels.DefaultCellStyle.BackColor = lightColor;
        for (var i = 0; i < dChannels.ColumnCount; i++)
        {
            dChannels.Columns[i].HeaderCell.Style.Alignment =
DataGridViewContentAlignment.MiddleCenter;
            dChannels.Columns[i].Width = 100;
            dChannels.Columns[i].SortMode =
DataGridViewColumnSortMode.NotSortable;
        }
        dChannels.Columns[0].HeaderText = "Канал";
        dChannels.Columns[1].HeaderText = "Категорія";
        dChannels.Columns[2].HeaderText = "Елемент категорії";
        dChannels.Columns[3].HeaderText = "Охоплення";
        dChannels.Columns[4].HeaderText = "Конверсія";
        dChannels.Columns[5].HeaderText = "Розрахунковий час";
        channelsPanel.Controls.Add(dChannels);
    }
    private void _addMessage(string messageText)
    {
        textBox.Text += messageText + "\r\n";
    }
    private void _openPopularityDialog(object sender, EventArgs e)
    {
        var fileDialog = new OpenFileDialog();
```

```
    if (fileDialog.ShowDialog() == DialogResult.OK)
    {
        fpPopularity.Text = fileDialog.FileName;
    }
}
private void _openChannelsDialog(object sender, EventArgs e)
{
    var fileDialog = new OpenFileDialog();
    if (fileDialog.ShowDialog() == DialogResult.OK)
    {
        fpChannels.Text = fileDialog.FileName;
    }
}
private void _addPopularity(object sender, EventArgs e)
{
    try
    {
        var popularity = new CustomerStatistics(fpPopularity.Text);
        var lastElement = dPopularity.RowCount - 1;
        dPopularity.RowCount += popularity.Count;
        for (var i = 0; i < popularity.Count; i++)
        {
            dPopularity[0, i + lastElement].Value = popularity.Name;
            dPopularity[1, i + lastElement].Value = popularity[i].Element;
            dPopularity[2, i + lastElement].Value = popularity[i].Value;
        }
        goodsPopularity.Add(popularity);
    }
    catch (Exception ex)
    {
```

```
            MessageBox.Show(ex.Message);
        }
    }
    private void _addChannel(object sender, EventArgs e)
    {
        try
        {
            var channel = new Channel(fpChannels.Text, AuxiliaryFunctions.
            testAgression);
            var lastElement = dChannels.RowCount - 1;
            var rowCount = channel.TotalElementsCount;
            dChannels.RowCount += rowCount;
            var k = 0;
            for (var i = 0; i < channel.Count; i++)
            {
                for (var j = 0; j < channel.CategoryCount(i); j++)
                {
                    dChannels[0, k + lastElement].Value = channel.Name;
                    dChannels[1, k + lastElement].Value = channel.CategoryName(i);
                    dChannels[2, k + lastElement].Value = channel.Penetration[i][j].Element;
                    dChannels[3, k + lastElement].Value = channel.Penetration[i][j].Value;
                    dChannels[4, k + lastElement].Value = channel.Conversion[i][j].Value;
                    k++;
                }
            }
            channels.Add(channel);
```

```
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
        private void _start(object sender, EventArgs e)
        {
            //try
            // {
            var time = Convert.ToDouble(tTime.Text);
            var budget = Convert.ToDouble(tBudget.Text);
            var bestPromotion = new BestPromotion(goodsPopularity, channels,
time, budget);
            var answer = bestPromotion.BestStrategy();
            for (var i = 0; i < answer.Count; i++)
                dChannels[5, i].Value = answer[i];
            // }
            // catch(Exception ex)
            // {
            // MessageBox.Show(ex.Message);
            // }
        }
        private void _formLoad(object sender, EventArgs e)
        {
            CenterToScreen();
        }
    }
}
```